**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)

Студенка Кузенкова Елизавета группы 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

class Lab8\_1

{

public static void Main(string[] args)

{

var app = new Lab8\_1();

app.DoWork(args);

}

private void DoWork(string[] args)

{

using (StreamWriter sw = new StreamWriter("output.txt"))

{

string[] stdin = File.ReadAllLines("input.txt");

var ts = new SortedSet<long>();

for (int i = 1; i < stdin.Length; i++)

{

var key = long.Parse(stdin[i].Split(' ')[1]);

switch (stdin[i][0])

{

case 'A': ts.Add(key); break;

case 'D': ts.Remove(key); break;

case '?':

if (ts.Contains(key))

sw.WriteLine("Y");

else

sw.WriteLine("N");

break;

default: break;

}

}

}

}

}

## Бенчмарк к задаче 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 1.265 | 93425664 | 11189636 | 501237 |
| 1 | OK | 0.031 | 11558912 | 43 | 9 |
| 2 | OK | 0.031 | 11165696 | 8 | 3 |
| 3 | OK | 0.031 | 11579392 | 51 | 12 |
| 4 | OK | 0.031 | 11444224 | 542 | 99 |
| 5 | OK | 0.046 | 11522048 | 618 | 54 |
| 6 | OK | 0.031 | 11608064 | 5451 | 1038 |
| 7 | OK | 0.046 | 11632640 | 6436 | 957 |
| 8 | OK | 0.031 | 11616256 | 13382 | 957 |
| 9 | OK | 0.046 | 11710464 | 22394 | 981 |
| 10 | OK | 0.031 | 11681792 | 7030 | 465 |
| 11 | OK | 0.031 | 11718656 | 7020 | 411 |
| 12 | OK | 0.031 | 13692928 | 63829 | 10002 |
| 13 | OK | 0.031 | 14528512 | 80339 | 4947 |
| 14 | OK | 0.046 | 14491648 | 80203 | 5034 |
| 15 | OK | 0.109 | 24424448 | 545113 | 100323 |
| 16 | OK | 0.109 | 28209152 | 639485 | 99282 |
| 17 | OK | 0.125 | 28217344 | 738870 | 99558 |
| 18 | OK | 0.156 | 28200960 | 1338668 | 99636 |
| 19 | OK | 0.203 | 28917760 | 2237627 | 99540 |
| 20 | OK | 0.171 | 28708864 | 903052 | 50202 |
| 21 | OK | 0.156 | 28717056 | 902843 | 49536 |
| 22 | OK | 0.312 | 47919104 | 2725205 | 501237 |
| 23 | OK | 0.359 | 53354496 | 3196877 | 499713 |
| 24 | OK | 0.406 | 53370880 | 3694712 | 501051 |
| 25 | OK | 0.890 | 57753600 | 6694340 | 500355 |
| 26 | OK | 1.265 | 75386880 | 11189636 | 500040 |
| 27 | OK | 0.765 | 62967808 | 4902931 | 249012 |
| 28 | OK | 0.765 | 62955520 | 4902757 | 250305 |
| 29 | OK | 0.875 | 88895488 | 9687139 | 300000 |
| 30 | OK | 0.890 | 88907776 | 9687570 | 300000 |
| 31 | OK | 0.765 | 80461824 | 8000008 | 300000 |
| 32 | OK | 0.828 | 93425664 | 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

class Lab8\_2

{

public static void Main(string[] args)

{

var app = new Lab8\_2();

app.DoWork(args);

}

private void DoWork(string[] args)

{

using (StreamWriter sw = new StreamWriter("output.txt"))

{

string[] stdin = File.ReadAllLines("input.txt");

var list = new LinkedList<string>();

var kv = new Dictionary<string, LinkedListNode<string>>();

for (int i = 1; i < stdin.Length; i++)

{

var arr = stdin[i].Split(' ');

var key = arr[1];

switch (arr[0])

{

case "put":

{

var value = arr[2];

if (kv.TryGetValue(key, out var node))

{

node.Value = value;

}

else

{

kv.Add(key, list.AddLast(value));

}

}

break;

case "get":

{

if (kv.TryGetValue(key, out var node))

sw.WriteLine(node.Value);

else

sw.WriteLine("<none>");

} break;

case "prev":

{

if (kv.TryGetValue(key, out var node)

&& node.Previous != null)

sw.WriteLine(node.Previous.Value);

else

sw.WriteLine("<none>");

}

break;

case "next":

{

if (kv.TryGetValue(key, out var node) && node.Next != null)

sw.WriteLine(node.Next.Value);

else

sw.WriteLine("<none>");

}

break;

case "delete":

{

if (kv.TryGetValue(key, out var node))

{

kv.Remove(key);

list.Remove(node);

}

} break;

default:

throw new NotImplementedException();

}

}

}

}

}

## Бенчмарк к задаче 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 1.593 | 231190528 | 23499808 | 10303658 |
| 1 | OK | 0.015 | 10924032 | 158 | 26 |
| 2 | OK | 0.031 | 10883072 | 12 | 8 |
| 3 | OK | 0.046 | 10895360 | 25 | 5 |
| 4 | OK | 0.031 | 10952704 | 25 | 8 |
| 5 | OK | 0.046 | 10952704 | 82 | 20 |
| 6 | OK | 0.031 | 10797056 | 1200 | 504 |
| 7 | OK | 0.046 | 10792960 | 1562 | 564 |
| 8 | OK | 0.031 | 11137024 | 12204 | 4617 |
| 9 | OK | 0.015 | 11067392 | 12058 | 4340 |
| 10 | OK | 0.140 | 27553792 | 960183 | 395964 |
| 11 | OK | 0.109 | 27787264 | 1318345 | 765350 |
| 12 | OK | 0.093 | 27623424 | 1420595 | 880052 |
| 13 | OK | 0.125 | 27594752 | 1079934 | 395020 |
| 14 | OK | 0.093 | 27656192 | 840022 | 332970 |
| 15 | OK | 0.109 | 27586560 | 1223121 | 889998 |
| 16 | OK | 0.156 | 32616448 | 3120970 | 486100 |
| 17 | OK | 0.140 | 32636928 | 3123298 | 486652 |
| 18 | OK | 0.156 | 32645120 | 3122193 | 479024 |
| 19 | OK | 0.093 | 27590656 | 900630 | 420456 |
| 20 | OK | 0.156 | 32600064 | 3121195 | 486718 |
| 21 | OK | 0.296 | 55656448 | 4199992 | 8 |
| 22 | OK | 0.265 | 54333440 | 4099993 | 8 |
| 23 | OK | 0.265 | 53665792 | 3999994 | 8 |
| 24 | OK | 0.265 | 53669888 | 3899995 | 8 |
| 25 | OK | 0.250 | 52162560 | 3799996 | 8 |
| 26 | OK | 0.234 | 51343360 | 3699997 | 8 |
| 27 | OK | 0.234 | 50823168 | 3599998 | 8 |
| 28 | OK | 0.250 | 50814976 | 3499999 | 8 |
| 29 | OK | 0.234 | 49565696 | 3400000 | 8 |
| 30 | OK | 0.265 | 52895744 | 3300001 | 8 |
| 31 | OK | 0.328 | 52711424 | 5399043 | 1973124 |
| 32 | OK | 0.265 | 52715520 | 4200443 | 1669405 |
| 33 | OK | 0.328 | 52723712 | 6099290 | 4429770 |
| 34 | OK | 0.671 | 96858112 | 15598672 | 2589784 |
| 35 | OK | 0.687 | 96739328 | 15589269 | 2586758 |
| 36 | OK | 0.640 | 97587200 | 15603830 | 2398360 |
| 37 | OK | 0.281 | 52723712 | 4499616 | 2110630 |
| 38 | OK | 0.687 | 96874496 | 15603381 | 2583188 |
| 39 | OK | 1.375 | 212168704 | 20999992 | 8 |
| 40 | OK | 1.390 | 204980224 | 20499993 | 8 |
| 41 | OK | 1.421 | 206655488 | 19999994 | 8 |
| 42 | OK | 1.343 | 206598144 | 19499995 | 8 |
| 43 | OK | 1.593 | 202289152 | 18999996 | 8 |
| 44 | OK | 1.328 | 200146944 | 18499997 | 8 |
| 45 | OK | 1.312 | 201854976 | 17999998 | 8 |
| 46 | OK | 1.328 | 199032832 | 17499999 | 8 |
| 47 | OK | 1.296 | 195747840 | 17000000 | 8 |
| 48 | OK | 1.546 | 192532480 | 16500001 | 8 |
| 49 | OK | 1.125 | 148766720 | 18500008 | 5499986 |
| 50 | OK | 1.453 | 231190528 | 23499808 | 220 |
| 51 | OK | 0.453 | 72785920 | 13500208 | 10303658 |
| 52 | OK | 0.859 | 100110336 | 15500008 | 8799944 |
| 53 | OK | 1.375 | 204312576 | 21500008 | 2200000 |
| 54 | OK | 1.140 | 148676608 | 18500008 | 5500000 |
| 55 | OK | 1.453 | 231178240 | 23499808 | 220 |
| 56 | OK | 0.468 | 75517952 | 13500208 | 10300130 |
| 57 | OK | 0.859 | 100093952 | 15500008 | 8799958 |
| 58 | OK | 1.359 | 204345344 | 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

public class HashTable

{

private long \_tableSize;

private long[] \_table;

public HashTable(long size)

{

\_tableSize = size;

\_table = new long[size];

for (int i = 0; i < size; i++)

\_table[i] = -1;

}

public bool TryInsert(long key)

{

long hash = this.GetHash(key);

long hash2 = this.GetHash2(key);

while (\_table[hash] != -1 && \_table[hash] != key)

{

hash = (hash + hash2) % \_tableSize;

hash2++;

}

if (\_table[hash] == key)

return false;

\_table[hash] = key;

return true;

}

private long GetHash(long key)

{

return Math.Abs(unchecked((int)((long)(key \* 47))) ^ (int)((key \* 31) >> 32)) % \_tableSize;

}

private long GetHash2(long key)

{

return Math.Abs(unchecked((int)((long)(key \* 113))) ^ (int)((key \* 97) >> 32)) % (\_tableSize - 1) + 1;

}

}

class Lab8\_3

{

public static void Main(string[] args)

{

var app = new Lab8\_3();

app.DoWork(args);

}

private void DoWork(string[] args)

{

using (StreamWriter sw = new StreamWriter("output.txt"))

{

string[] stdin = File.ReadAllLines("input.txt");

var arr1 = stdin[0].Split(' ').Select(s => long.Parse(s)).ToArray();

var arr2 = stdin[1].Split(' ').Select(s => long.Parse(s)).ToArray();

var n = arr1[0];

var x = arr1[1];

var a = arr1[2];

var b = arr1[3];

var Ac = arr2[0];

var Bc = arr2[1];

var Ad = arr2[2];

var Bd = arr2[3];

HashTable hashTable = new HashTable(n \* 2);

for (int i = 0; i < n; i++)

{

if (hashTable.TryInsert(x))

{

a = (a + Ad) % 1000;

b = (b + Bd) % 1000000000000000;

}

else

{

a = (a + Ac) % 1000;

b = (b + Bc) % 1000000000000000;

}

x = (x \* a + b) % 1000000000000000;

}

sw.WriteLine($"{x} {a} {b}");

}

}

}

## Бенчмарк к задаче 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 2.078 | 170520576 | 87 | 37 |
| 1 | OK | 0.031 | 10153984 | 18 | 7 |
| 2 | OK | 0.031 | 10207232 | 19 | 7 |
| 3 | OK | 0.031 | 10174464 | 21 | 7 |
| 4 | OK | 0.031 | 10182656 | 21 | 7 |
| 5 | OK | 0.031 | 10186752 | 21 | 7 |
| 6 | OK | 0.031 | 10248192 | 21 | 15 |
| 7 | OK | 0.015 | 10194944 | 21 | 7 |
| 8 | OK | 0.062 | 10174464 | 21 | 9 |
| 9 | OK | 0.015 | 10203136 | 21 | 9 |
| 10 | OK | 0.015 | 10190848 | 30 | 28 |
| 11 | OK | 0.015 | 10190848 | 30 | 28 |
| 12 | OK | 0.031 | 10162176 | 35 | 35 |
| 13 | OK | 0.046 | 10182656 | 47 | 32 |
| 14 | OK | 0.031 | 10190848 | 63 | 35 |
| 15 | OK | 0.031 | 10182656 | 81 | 37 |
| 16 | OK | 0.046 | 10309632 | 82 | 37 |
| 17 | OK | 0.015 | 11755520 | 23 | 7 |
| 18 | OK | 0.031 | 11776000 | 23 | 7 |
| 19 | OK | 0.015 | 11788288 | 23 | 7 |
| 20 | OK | 0.031 | 11800576 | 23 | 21 |
| 21 | OK | 0.031 | 11829248 | 23 | 7 |
| 22 | OK | 0.031 | 11784192 | 23 | 9 |
| 23 | OK | 0.031 | 11780096 | 23 | 9 |
| 24 | OK | 0.046 | 11894784 | 32 | 30 |
| 25 | OK | 0.046 | 11804672 | 32 | 30 |
| 26 | OK | 0.031 | 11796480 | 37 | 35 |
| 27 | OK | 0.031 | 11763712 | 51 | 35 |
| 28 | OK | 0.031 | 11796480 | 64 | 34 |
| 29 | OK | 0.046 | 11755520 | 84 | 37 |
| 30 | OK | 0.031 | 11776000 | 84 | 36 |
| 31 | OK | 0.078 | 26189824 | 24 | 7 |
| 32 | OK | 0.078 | 26202112 | 24 | 7 |
| 33 | OK | 0.078 | 26198016 | 24 | 7 |
| 34 | OK | 0.187 | 26148864 | 24 | 24 |
| 35 | OK | 0.078 | 26189824 | 24 | 7 |
| 36 | OK | 0.062 | 26189824 | 24 | 9 |
| 37 | OK | 0.078 | 26210304 | 24 | 9 |
| 38 | OK | 0.171 | 26198016 | 33 | 16 |
| 39 | OK | 0.125 | 26169344 | 33 | 30 |
| 40 | OK | 0.203 | 26230784 | 38 | 35 |
| 41 | OK | 0.187 | 26193920 | 52 | 35 |
| 42 | OK | 0.187 | 26173440 | 66 | 35 |
| 43 | OK | 0.187 | 26230784 | 84 | 37 |
| 44 | OK | 0.171 | 26185728 | 85 | 37 |
| 45 | OK | 0.500 | 170455040 | 25 | 7 |
| 46 | OK | 0.531 | 170491904 | 25 | 7 |
| 47 | OK | 0.546 | 170520576 | 25 | 7 |
| 48 | OK | 1.984 | 170491904 | 25 | 27 |
| 49 | OK | 0.484 | 170450944 | 25 | 7 |
| 50 | OK | 0.484 | 170446848 | 25 | 9 |
| 51 | OK | 0.484 | 170459136 | 25 | 9 |
| 52 | OK | 1.625 | 170430464 | 34 | 16 |
| 53 | OK | 1.359 | 170455040 | 34 | 30 |
| 54 | OK | 1.984 | 170459136 | 39 | 35 |
| 55 | OK | 1.968 | 170450944 | 51 | 35 |
| 56 | OK | 2.015 | 170467328 | 67 | 35 |
| 57 | OK | 2.015 | 170508288 | 87 | 37 |
| 58 | OK | 2.031 | 170426368 | 87 | 37 |
| 59 | OK | 2.015 | 170487808 | 87 | 35 |
| 60 | OK | 2.000 | 170467328 | 86 | 37 |
| 61 | OK | 2.015 | 170442752 | 87 | 37 |
| 62 | OK | 2.000 | 170459136 | 86 | 37 |
| 63 | OK | 2.031 | 170483712 | 86 | 37 |
| 64 | OK | 2.031 | 170483712 | 86 | 37 |
| 65 | OK | 2.015 | 170455040 | 87 | 37 |
| 66 | OK | 2.000 | 170446848 | 85 | 35 |
| 67 | OK | 2.031 | 170446848 | 85 | 36 |
| 68 | OK | 2.078 | 170442752 | 87 | 36 |