Санкт-Петербургский национальный исследовательский университет

Информационных технологий, механики и оптики

Отчет по решению задач первой недели
По курсу «Алгоритмы и структуры данных»

на Openedu

Выполнил: Сыроватский Павел Валентинович

Группа P3218

Санкт-Петербург

2019

Множество

2.0 из 2.0 баллов (оценивается)

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 $5 \cdot 10^5$. 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

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

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Openedu.Week8
{
    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;
                    }
                }
            }
        }
    }
}
```

Результат выполнения первой задачи

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.750	26288128	1978102	1277538
1	OK	0.031	9031680	22	17
2	OK	0.062	8953856	20	38
3	OK	0.031	8998912	41	15
4	OK	0.078	13709312	204081	21587
5	OK	0.062	14024704	412716	21559
6	OK	0.078	14004224	412714	12243
7	OK	0.375	17354752	498728	612555
8	OK	0.390	17723392	1008458	612906
9	OK	0.328	17715200	1008832	341682
10	OK	0.484	18759680	471365	861755
11	OK	0.515	20160512	953290	859761
12	OK	0.406	20246528	953404	548738
13	OK	0.093	12963840	197660	51796
14	OK	0.093	13565952	399789	51761
15	OK	0.078	13836288	399826	29610
16	OK	0.546	19644416	511344	947660
17	OK	0.546	20410368	1034328	951787
18	OK	0.421	21270528	1034511	608920
19	OK	0.187	16142336	384717	274370
20	OK	0.234	16449536	777782	274601
21	OK	0.171	16408576	778270	152655
22	OK	0.156	12550144	219786	228823
23	OK	0.171	12738560	444845	228627
24	OK	0.140	12640256	444580	136297
25	OK	0.109	19619840	452007	84006
26	OK	0.140	19869696	914248	84077

Прошитый ассоциативный массив

ЭТОТ ЭЛЕМЕНТ КУРСА ОЦЕНИВАЕТСЯ КАК 'ЛАБОРАТОРНАЯ РАБОТА'

ВЕС: 2.0

Добавить страницу в мои закладки

Прошитый ассоциативный массив

2.0 из 2.0 баллов (оценивается)

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 $5 \cdot 10^5$. 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 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 `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.

Реализация программы на C++

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Openedu.Week8
{
    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();
                    }
                }
            }
        }
    }
}
```

Результат решения задачи

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.437	231084032	23499808	10303658
1	OK	0.046	10915840	158	26
2	OK	0.031	10862592	12	8
3	OK	0.031	10899456	25	5
4	OK	0.015	10952704	25	8
5	OK	0.031	10956800	82	20
6	OK	0.031	10956800	1200	504
7	OK	0.031	10993664	1562	564
8	OK	0.031	11329536	12204	4617
9	OK	0.031	11272192	12058	4340
10	OK	0.109	27787264	960183	395964
11	OK	0.125	27815936	1318345	765350
12	OK	0.109	27807744	1420595	880052
13	OK	0.125	27615232	1079934	395020
14	OK	0.093	27566080	840022	332970
15	OK	0.125	27561984	1223121	889998
16	OK	0.140	32538624	3120970	486100
17	OK	0.156	32624640	3123298	486652
18	OK	0.140	32628736	3122193	479024
19	OK	0.093	27578368	900630	420456
20	OK	0.156	32567296	3121195	486718
21	OK	0.265	55635968	4199992	8

Почти интерактивная хеш-таблица

ЭТОТ ЭЛЕМЕНТ КУРСА ОЦЕНИВАЕТСЯ КАК 'ЛАБОРАТОРНАЯ РАБОТА'

ВЕС: 2.0

Добавить страницу в мои закладки

Почти интерактивная хеш-таблица

2.0 из 2.0 баллов (оценивается)

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 $[0;1015-1]$. 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:

- $1 \le N \le 107$;
- $0 \le X < 1015$;
- $0 \le A < 103$;
- $0 \le B < 1015$.

Требуется N раз выполнить следующую последовательность операций:

- Если X содержится в таблице, то установить $A \leftarrow (A+AC) \bmod 103$, $B \leftarrow (B+BC) \bmod 1015$.
- Если X не содержится в таблице, то добавить X в таблицу и установить $A \leftarrow (A+AD) \bmod 103$, $B \leftarrow (B+BD) \bmod 1015$.
- Установить $X \leftarrow (X \cdot A+B) \bmod 1015$.

Начальные значения X, A и B, а также N, AC, BC, AD и BD даны во входном файле. Выведите значения X, A и 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 $0 \le AC,AD < 103$, $0 \le BC,BD < 1015$.

Реализация программы на C++

```
#include "edx-io.hpp"
#include <map>
#include <string>
using namespace std;

//Хеш-функция
long get_hash(long long value, long ht_size)
{
    return abs((value * 47) ^ (value * 31)) % ht_size;
}
//Возвращает true если элемент вставлен, false - если такой элемент уже был добавлен
bool insert_into_ht(long long*& ht, long long value, long ht_size)
{
    //Вычисляем хеш
    long hash = get_hash(value, ht_size);
    //Пока не наткнёмся на свободную ячейку или ячейку с этим же значением двигаемся
вперёд на одну ячейку
    while (ht[hash] != -1 && ht[hash] != value)
    {
        //Зацикливаем массив
        if (++hash == ht_size)
        {
            hash = 0;
        }
    }
    if (ht[hash] == value)
    {
        return false;
    }
    else
    {
        ht[hash] = value;
        return true;
    }
}

int main()
{
    long N;
    int A, Ac, Ad;
    long long X, B, Bc, Bd;

    io >> 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_into_ht(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;
```

```
    }

    io << X << " " << A << " " << B;

    return 0;
}
```

Результат решения задачи

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		2.140	170549248	87	37
1	OK	0.031	10391552	18	7
2	OK	0.046	10432512	19	7
3	OK	0.015	10428416	21	7
4	OK	0.015	10416128	21	7
5	OK	0.031	10383360	21	7
6	OK	0.031	10436608	21	15
7	OK	0.031	10399744	21	7
8	OK	0.031	10391552	21	9
9	OK	0.031	10399744	21	9
10	OK	0.031	10457088	30	28
11	OK	0.031	10387456	30	28
12	OK	0.046	10412032	35	35
13	OK	0.031	10199040	47	32
14	OK	0.031	10186752	63	35
15	OK	0.046	10244096	81	37
16	OK	0.031	10223616	82	37
17	OK	0.015	11776000	23	7
18	OK	0.031	11767808	23	7
19	OK	0.031	11784192	23	7
20	OK	0.031	11800576	23	21
21	OK	0.031	11780096	23	7
22	OK	0.031	11874304	23	9
23	OK	0.015	11837440	23	9