Санкт-Петербургский Национальный Исследовательский Университет
Информационных Технологий, Механики и Оптики

Лабораторная работа №1

по дисциплине

«Алгоритмы и структуры данных»

Выполнил: Студент группы P3217

Сергачев Данила Дмитриевич

Преподаватели:

Романов Алексей Андреевич и

Волчек Дмитрий Геннадьевич

Санкт-Петербург

2019 г

Задание №1

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».

Решение

```
package week8;

import mooc.EdxIO;

import java.util.LinkedList;

public class task1 {
   public static void main(String[] args) {
      try (EdxIO io = EdxIO.create()) {
         long n = io.nextLong();

         StringBuilder st = new StringBuilder();
         Set set = new Set(65537);

         for(long i = 0; i < n; i++){
            switch (io.nextChar()){
               case 'A' : {
                  set.add(io.nextLong());

                  break;
               } case 'D': {
                  set.delete(io.nextLong());

                  break;
               } case '?' : {
                  st.append(set.find(io.nextLong(), false) ? "Y\n" : "N\n") ;

                  break;
               }
            }
         }

         io.print(st);
      }
   }
```

```
static class Set
{
   private LinkedList<Long>[] _array;
   private int _divider;

   public Set(int primeNumber)
   {
      _divider = primeNumber;
      _array = new LinkedList[_divider * 2 - 1];
   }

   private long Hash(long number)
   {
      return number % _divider + 65536;
   }

   private boolean isListExists(long number)
   {
      return _array[(int) Hash(number)] == null ? false : true;
   }

   public void add(long number)
   {
      if (!isListExists(number))
      {
         _array[(int) Hash(number)] = new LinkedList<Long>();
         _array[(int) Hash(number)].addFirst(number);
      }
      else
      {
         if (!find(number,true))
         {
            _array[(int) Hash(number)].addFirst(number);
         }
      }

   }

   public boolean find(final long number, boolean isListChecked)
   {
      if (isListChecked || isListExists(number))
      {
         return _array[(int) Hash(number)].contains(number);
      }
      return false;
   }

   public void delete(long number)
   {
      if (find(number, false))
      {
         _array[(int) Hash(number)].removeFirstOccurrence(number);
      }
   }
```

```
    }
}
```

Результат работы

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.546	61063168	11189636	501237
1	OK	0.109	22151168	43	9
2	OK	0.125	22200320	8	3
3	OK	0.109	22188032	51	12
4	OK	0.109	22147072	542	99
5	OK	0.125	22192128	618	54
6	OK	0.140	22306816	5451	1038
7	OK	0.125	22310912	6436	957
8	OK	0.140	22355968	13382	957
9	OK	0.109	22462464	22394	981
10	OK	0.140	22413312	7030	465

Задание №2

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:

Решение

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

namespace week8
{
    class task2
    {
        public static void Main(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 работы

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.453	231477248	23499808	10303658
1	OK	0.015	10747904	158	26
2	OK	0.031	10715136	12	8
3	OK	0.031	10797056	25	5
4	OK	0.031	10772480	25	8
5	OK	0.046	10764288	82	20
6	OK	0.031	10780672	1200	504
7	OK	0.015	10821632	1562	564
8	OK	0.031	11182080	12204	4617
9	OK	0.031	11132928	12058	4340
10	OK	0.093	27623424	960183	395964
11	OK	0.093	27635712	1318345	765350

Задание №3

Почти интерактивная хеш-таблица

2.0 из 2.0 баллов (оценивается)

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 $[0; 10^{15} - 1]$. 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:

- $1 \le N \le 10^7$;
- $0 \le X < 10^{15}$;
- $0 \le A < 10^3$;
- $0 \le B < 10^{15}$.

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.

Решение

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Globalization;

namespace week8
{
    public class Task3
    {
        public static void Main(string[] args)
        {
            using (StreamReader streamReader = new StreamReader("input.txt"))
            using (StreamWriter streamWriter = new StreamWriter("output.txt"))
            {
                string[] str = streamReader.ReadLine().Split(' ');
                int n = int.Parse(str[0]);
                long x = long.Parse(str[1]);
                int a = int.Parse(str[2]);
                long b = long.Parse(str[3]);
                str = streamReader.ReadLine().Split(' ');
                int ac = int.Parse(str[0]);
                long bc = long.Parse(str[1]);
                int ad = int.Parse(str[2]);
                long bd = long.Parse(str[3]);

                HashTable hashTable = new HashTable(n * 2);
                for (int i = 0; i < n; i++)
                {
                    if (hashTable.Insert(x))
                    {
                        a = (a + ad) % 1000;
                        b = (b + bd) % 1000000000000000;
                    }
                    else
                    {
                        a = (a + ac) % 1000;
                        b = (b + bc) % 1000000000000000;
                    }
                    x = (x * a + b) % 1000000000000000;
                }
```

```
                streamWriter.WriteLine("{0} {1} {2}", x, a, b);
            }
        }

        public class HashTable
        {
            private int tableSize;
            private long[] table;

            public HashTable(int size)
            {
                tableSize = size;
                table = new long[size];
                for (int i = 0; i < size; i++)
                    table[i] = -1;
            }

            public bool Insert(long key)
            {
                int hash = GetHash(key);
                int hash2 = 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 int GetHash(long key)
            {
                return Math.Abs(unchecked((int)((long)(key * 47))) ^ (int)((key * 31) >>
32)) % tableSize;
            }
            private int GetHash2(long key)
            {
                return Math.Abs(unchecked((int)((long)(key * 113))) ^ (int)((key * 97) >>
32)) % (tableSize - 1) + 1;
            }
        }
    }
}
```

Результат работы

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		2.125	170516480	87	37
1	OK	0.031	10289152	18	7
2	OK	0.031	10174464	19	7
3	OK	0.031	10268672	21	7
4	OK	0.015	10240000	21	7
5	OK	0.031	10272768	21	7
6	OK	0.031	10203136	21	15
7	OK	0.031	10207232	21	7
8	OK	0.031	10248192	21	9
9	OK	0.031	10240000	21	9