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

Задачи восьмой недели

по курсу «Алгоритмы и структуры данных»

на Openedu

Выполнил: студент группы P3218

Артамонов Александр Владимирович

Санкт-Петербург, 2019г

Задача 1. Множество

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.

Пример

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

Код программы (C++)

```
//Удаляет k-тый элемент списка list1
auto delete_list_elem(list<long long>& list1, int k)
{
	list<long long>::iterator it = list1.begin();
	std::advance(it, k); // <-- advance итерирует переданный итератор на k позиций
	if (it != list1.end())
	{
		return list1.erase(it); // <--- Вернет итератор на k+1 элемент, перед it нет *
	}
	return it;
}

//Возвращает позицию элемента со значением value, если он есть, в обратном случае возвращает -1
int find_list_elem(list<long long> list1, long long value) {
	int counter = 0;
	int size = list1.size();
	for (long long number : list1) {
		if (number == value) {
			break;
		}
		else {
			counter++;
		}
	}
	if (counter == size) {
		return -1;
	}
	return counter;
}

int main() {
	long N;
	io >> N;

	//Создаём массив листов - закрытая адресация
	list<long long>* ht = new list<long long>[N];

	char command;
	long long element;
	long hash;
	bool is_in_list;

for (long i = 0; i < N; i++) {
		io >> command >> element;
		hash = abs(element) % N;
		switch (command)
		{
		case 'A':
			//Если такое значение уже есть в списке, то ничего не делаем, если нет, то вставляем в начало списка
			is_in_list = false;
			for (long long number : ht[hash]) {
				if (number == element) {
					is_in_list = true;
				}
			}
			if (!is_in_list) {
				ht[hash].push_front(element);
			}
			break;
```

```
                case 'D':
                        //Если такое значение есть в списке - удаляем его
                        if (!ht[hash].empty()) {
                                int pos = find_list_elem(ht[hash], element);
                                if (pos != -1) {
                                        delete_list_elem(ht[hash], pos);
                                }
                        }
                        break;
                default:
                        if (!ht[hash].empty()) {
                                int pos = find_list_elem(ht[hash], element);
                                if (pos != -1) {
                                        io << "Y\n";
                                        break;
                                }
                        }
                        io << "N\n";
                        break;
                }
        }

        return 0;
}
```

Бенчмарк (задача 1)

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.750	51511296	11189636	501237
1	OK	0.000	2220032	43	9

Задача 2. Прошитый ассоциативный массив

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 putдолжен 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.

Формат выходного файла

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

Пример

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	

Код программы (C++)

```
//Возвращает позицию предыдущего элемента для pos, если он есть, и -1 в обратном случае
int get_prev_pos(string* pos_string, int pos) {
	pos--;
	while (pos > 0 && pos_string[pos].empty())
	{
		pos--;
	}
	if (!pos_string[pos].empty() && pos > -1) {
		return pos;
	}
	else {
		return -1;
	}
}
//Возвращет позицию следующего элемента для pos, если он есть, и -1 в обратном случае
int get_next_pos(string* pos_string, int pos, long keys_counter) {
	pos++;
	while (pos < keys_counter && pos_string[pos].empty())
	{
		pos++;
	}
	if (!pos_string[pos].empty() && pos > -1 && pos < keys_counter) {
		return pos;
	}
	else {
		return -1;
	}
}
```

```
int main() {
	long N;
	io >> N;
	//Ассоциативный массив
	map<string, string> ht;
	//Массив, хранящий для каждого ключа порядок добавления в ассоциативный массив
	map<string, long> keys_pos;
	//Массив, хранящий ключи в остортированном по добавлению порядке
	string* pos_string = new string[N];
	//Счётчик введённых ключей
	long key_counter = 0;
	int pos, prev_pos;
	string command, key, element;

	for (long i = 0; i < N; i++) {
		io >> command >> key;
		switch (command[0])
		{
		case 'p':
			//put
			if (command[1] == 'u') {
				io >> element;
				if (!ht[key].empty()) {
					ht[key].assign(element);
				}
				else {
					ht[key].assign(element);
					keys_pos[key] = key_counter;
					pos_string[key_counter++] = key;
				}
				break;
			}
			//prev
			else {
				pos = keys_pos[key];
				prev_pos = get_prev_pos(pos_string, pos);
				if (prev_pos != -1) {
					io << ht[pos_string[prev_pos]] << "\n";
				}
				else
				{
					io << "<none>\n";
				}
				break;
			}
		case 'n':
			pos = keys_pos[key];
			if (pos <= 0 && pos_string[pos] != key) {
				prev_pos = -1;
			}
			else {
				prev_pos = get_next_pos(pos_string, pos, key_counter);
			}
			if (prev_pos != -1) {
				io << ht[pos_string[prev_pos]] << "\n";
			}
			else
			{
				io << "<none>\n";
			}
			break;
		case 'g':
			element = ht[key];
			if (!element.empty()) {
				io << element << "\n";
```

```
                        }
                        else {
                                io << "<none>\n";
                        }
                        break;
                case 'd':
                        ht[key].assign("");
                        if (keys_pos[key] == -1 || (keys_pos[key] == 0 && pos_string[0] !=
key)) {

                        }
                        else {
                                pos_string[keys_pos[key]] = "";
                        }
                        keys_pos[key] = -1;
                        break;
                default:
                        break;
                }
        }

        return 0;
}
```

Бенчмарк (задача 2)

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		2.578	234369024	23499808	10303658
1	OK	0.031	2236416	158	26

Задача 3. 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:

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

Требуется N раз выполнить следующую последовательность операций:

- Если X содержится в таблице, то установить $A \leftarrow (A+AC) \bmod 10^3$, $B \leftarrow (B+BC) \bmod 10^{15}$.
- Если X не содержится в таблице, то добавить X в таблицу и установить $A \leftarrow (A+AD) \bmod 10^3$, $B \leftarrow (B+BD) \bmod 10^{15}$.
- Установить $X \leftarrow (X \cdot A+B) \bmod 10^{15}$.

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 $0 \le AC, AD < 10^3$, $0 \le BC, BD < 10^{15}$.

Формат выходного файла

Выведите значения X, A и B после окончания работы.

Пример

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

Код программы (C++)

```
//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;
	//Создаём массив в два раза большего размера
	long long* ht = new long long[N * 2];

	//-1 - обозначение для свободной ячейки
	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;
}
```

Бенчмарк (задача 3)

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		3.250	162238464	87	35
1	OK	0.000	2220032	18	5