Министерство науки и образования РФ

Федеральное государственное автономное образовательное

учреждение высшего профессионального образования

«Санкт-Петербургский государственный электротехнический

университет «ЛЭТИ» им. В. И. Ульянова (Ленина)»

(СПбГЭТУ «ЛЭТИ»)

Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

**Отчёт по заданию № 3**

**на тему: “Множества + последовательности”**

**по дисциплине**

**“Алгоритмы и структуры данных”**

**Вариант 44**

|  |  |
| --- | --- |
| Выполнили студенты гр.9308: | Дементьев Д.П., Ручкин Д.А. |
| Проверил: | Колинько П.Г. |

Санкт-Петербург, 2021 г.

Оглавление

[Введение 3](#_Toc72680555)

[1. Задание 3](#_Toc72680556)

[2. Формализация задания 3](#_Toc72680557)

[3. Описание контейнера 4](#_Toc72680558)

[4. Оценка временной сложности операций 5](#_Toc72680559)

[5. Примеры работы программы 6](#_Toc72680560)

[Вывод 9](#_Toc72680571)

[Список используемых источников 10](#_Toc72680572)

[Приложение 11](#_Toc72680573)

## Введение

* 1. Получить практические навыки работы со стандартной библиотекой шаблонов, с деревьями двоичного поиска и с последовательностями.

## 1. Задание

Реализовать индивидуальное задание темы «Множества + последовательности» в виде программы, используя свой контейнер для заданной структуры данных (хеш-таблицы или одного из вариантов ДДП), и доработать его для поддержки операций с последовательностями.

## 2. Формализация задания

**Мощность множества:** 52

**Что надо вычислить:** (A + B) ⊕ (C ∩ D) \ E

**Базовая структура данных:** АВЛ-дерево

**Реализуемые операции над последовательностью:**

ERASE — Из последовательности исключается часть, ограниченная порядковыми номерами от p1 до p2.

CONCAT — Вторая последовательность подсоединяется к концу первой, образуя ее продолжение.

CHANGE — Вторая последовательность заменяет элементы первой, начиная с заданной позиции p.

## 3. Описание контейнера

Контейнер *set\_seq* хранит множество в виде АВЛ-дерева и последовательность в виде вектора итераторов на узлы дерева. При обходе дерева получается упорядоченная последовательность ключей, а при обходе вектора — произвольная. Также каждый узел дерева имеет поле *duplicates* — количество дубликатов ключа.

Для работы с АВЛ-деревом был создан контейнер *tree*. Для доступа к элементам дерева был создан прямой итератор ввода *tree\_iterator*, который перемещается только вперед. Для него нужно определить операции сравнения (==, !=), разыменования (\*), инкремент (++). Для итератора чтения также необходимы функции *begin*() и *end*(), определяющие рабочий интервал значений.

Также для контейнера set\_seq необходимо создать итератор вставки. Итератор вставки создаётся из контейнера или одного из его итераторов, указывающих, где вставка происходит, если это ни в начале, ни в конце контейнера. Итераторы вставки удовлетворяют требованиям итераторов вывода. Для данного итератора определены операторы присвоения (=), разыменовывания (\*) и инкремента (++), причем все они фиктивные. Для поддержки итератора вставки нужно определить функцию вставки с сигнатурой *insert*(*where*, *data*), где *where* — итератор места вставки, а *data* — вставляемое значение. Функция должна возвращать итератор на вставленный элемент, чтобы обеспечивать вставку за константное время за счёт исключения необходимости поиска места вставки.

## 4. Оценка временной сложности операций

1. **Вставка (insert)**

Временная сложность для произвольной вставки составляет O(logn).

1. **Пересечение множеств (operator&=)**

Для пересечения по схеме слияния происходит не более 2\*(N1+N2)-1 сравнений, где N1 и N2 — размеры контейнеров, т.е временная сложность алгоритма O(n).

1. **Объединение множеств (operator+=)**

Для объединения по схеме слияния происходит не более 2\*(N1+N2)-1 сравнений, где N1 и N2 — размеры контейнеров, т.е временная сложность алгоритма O(n).

1. **Разность множеств (operator-=)**

Для разности по схеме слияния происходит не более 2\*(N1+N2)-1 сравнений, где N1 и N2 — размеры контейнеров, т.е временная сложность алгоритма O(n).

1. **Симметрическая разность множеств (operator^=)**

Для симметрической разности по схеме слияния происходит не более 2\*(N1+N2)-1 сравнений, где N1 и N2 — размеры контейнеров, т.е временная сложность алгоритма O(n).

1. **Вырезание части последовательности(ERASE)**

Временная сложность операции линейная O(n).

1. **Присоединение последовательностей (CONCAT)**

Временная сложность операции линейная O(n).

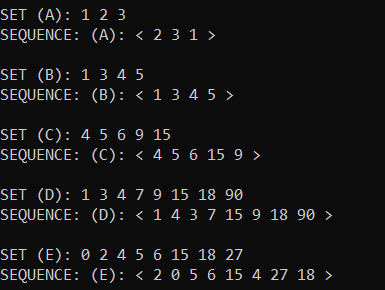
1. **Замена элементов последовательности(CHANGE)**

Временная сложность операции линейная O(n).

## 5. Примеры работы программы

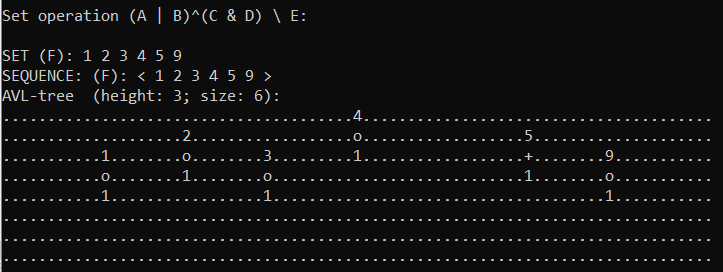
**1)** **Тестовый пример из программы**

Исходные множества и последовательности:



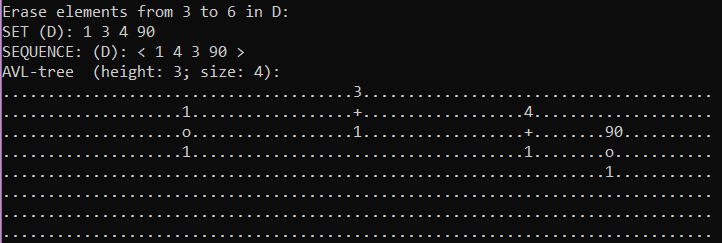
## *Рис.1. Исходные множества и последовательности*

Пример выполнения операций над множеством:



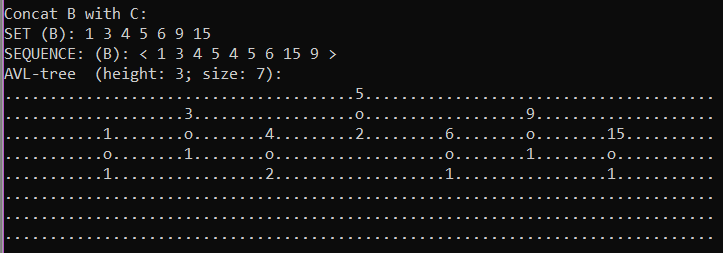
## *Рис.2. Операции над множеством*

Пример выполнения операции ERASE. Из последовательности D удалим элементы с 3 по 6.



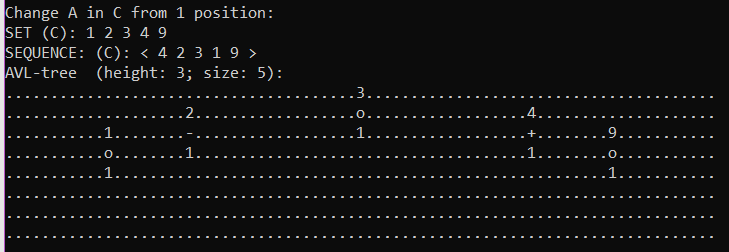
## *Рис.3. Операция ERASE*

Пример выполнения операции CONCAT. К концу последовательности B присоединим последовательность C.



## *Рис.4. Операция CONCAT*

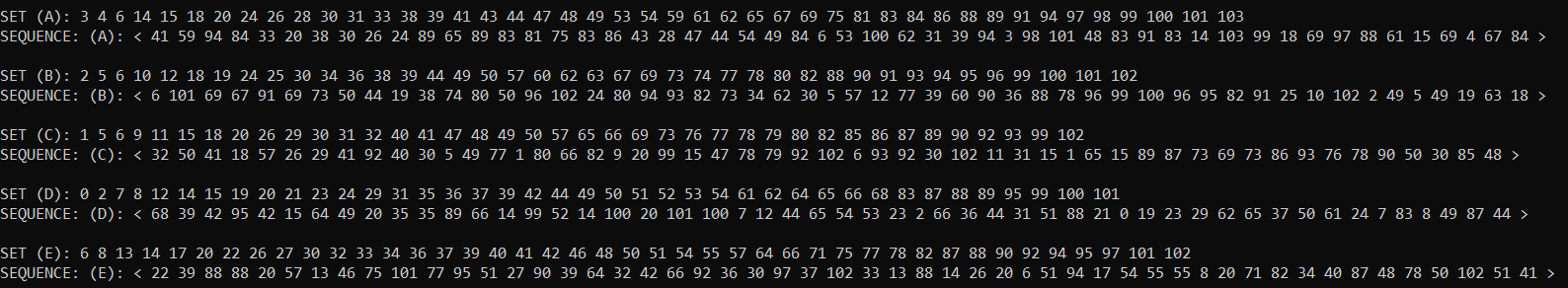
Пример выполнения операции CHANGE. В последовательности C, начиная с первой позиции, заменим элементы исходные элементами последовательности A.



## *Рис.5. Операция CHANGE*

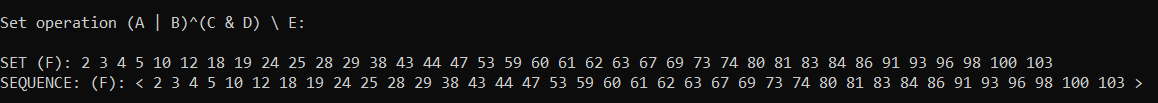
**2) Пример на случайных данных**

Сгенерированные множества и последовательности:



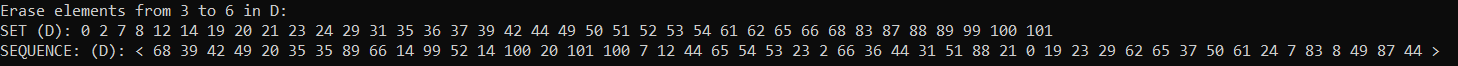
## *Рис.6. Случайно сгенерированные множества и последовательности*

Пример выполнения операций над множеством:



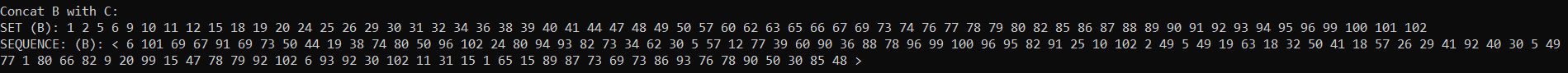
## *Рис.7. Операции над случайными множествами*

Пример выполнения операции ERASE. Из последовательности D удалим элементы с 3 по 6.



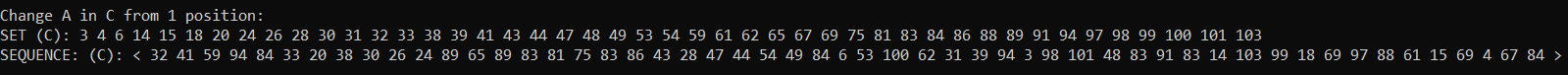
## *Рис.8. Операция ERASE*

Пример выполнения операции CONCAT. К концу последовательности B присоединим последовательность C.



## *Рис.9. Операция CONCAT*

Пример выполнения операции CHANGE. В последовательности C, начиная с первой позиции, заменим элементы исходные элементами последовательности A.



## *Рис.10. Операция CHANGE*

## Вывод

При выполнении данной работы получены практические навыки по реализации собственных контейнеров, работе с множествами и последовательностями.

## Список используемых источников

1. Колинько П.Г. Пользовательские контейнеры / Методические указания по дисциплине «Алгоритмы и структуры данных» - Санкт-Петербург: СПбГЭТУ «ЛЭТИ», 2020.
2. АВЛ-деревья. Хабр. URL: https://habr.com/ru/post/150732/ [дата обращения 17.05.2021]
3. Алгоритм set\_difference(). ВикиЧтение. URL: <https://it.wikireading.ru/36208/> [дата обращения 22.05.2021]
4. Итераторы вставки. КГСУ. URL: http://it.kgsu.ru/C\_STL/c\_stl103.html [дата обращения 21.05.2021]

## Приложение

screen.h — функции для работы с экраном

AVL.h — реализация АВЛ дерева

set\_seq.h — реализация контейнера

main.cpp — демонстрационная программа