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 40

Выполнил студент гр.9308: Аюпов Р.Н.

Проверил: Колинько П.Г.

Санкт-Петербург, 2020 г.

Оглавление

Введение

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: $(A \setminus B) \cup C \oplus D \cap E$

Базовая СД: 1-2д — это упрощённый вариант 2-3-4 дерева двоичного поиска с автобалансировкой. В нём используются только 2- и 3-узлы. От обычного ДДП 1-2-дерево отличается тем, что его узлы могут объединяться в пары, образуя узлы с двумя ключами и тремя возможными сыновьями. Такое дерево должно быть сбалансированным, если при подсчёте длины пути из корня в любой лист считать каждую пару за один узел. Фактические же длины путей могут различаться не более чем в два раза.

Реализуемые операции над последовательностью:

MERGE — Объединение двух упорядоченных последовательностей в третью с сохранением упорядоченности. От операции объединения множеств отличается только возможностью появления дубликатов ключей. Если исходные последовательности не упорядочены, можно после их слияния просто упорядочить результат. Исходный порядок ключей в последовательностях в результате не сохраняется.

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

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

3. Описание контейнера

Контейнер *mycont* хранит множество в виде 1-2-дерева и последовательность в виде вектора ключей. Обход дерева даёт упорядоченную последовательность ключей, а обход вектора — произвольную, что позволяет работать со структурой данных и как с множеством, и как с последовательностью.

Для доступа к элементам контейнера был создан прямой итератор ввода (чтения) *Iterator*. Такой итератор перемещается только вперед и поддерживает только чтение. Для него нужно определить операции сравнения (==, !=), разыменования (*), инкремент (++). Для итератора чтения также необходимы функции *begin*() и *end*(), определяющие рабочий интервал значений. Основной операцией является инкремент, и для того, чтобы он выполнялся за константное время, итератор хранит стек с путём от корня до текущего узла.

Для контейнера *mycont* также был реализован итератор вставки. Итератор вставки создаётся из контейнера и, возможно, 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 операции, кроме присвоения по ключу, 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. Итератор вставки хранит итератор чтения на вставленное значение, поддерживая тем самым возможную последовательность вставок.

4. Оценка временной сложности операций

Вставка (insert)

В контейнере *mycont* элемент вставляется и в дерево и в последовательность.

Вставка в дерево без указания места начала поиска имеет логарифмическое время O(logn). Возможна вставка за константное время O(1) если итератор вставки укажет для начала поиска места вставки на ключ, вставленный последним. Такая вставка допустима только для двуместных операций с множествами по схеме слияния.

Вставка в последовательность осуществляется за константное время O(1) (при условии, что контейнер не расширяется при добавлении в него элемента, если происходит перераспределение, то само перераспределение является линейным по всему размеру).

Поэтому в среднем временная сложность алгоритма вставки для операций над множествами по схемам слияния O(1), а для произвольной вставки сложность составляет O(logn).

Пересечение 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).

Объединение множеств (operator =)

Временная сложность операции O(n).

Разность множеств (operator-=)

Временная сложность операции O(n).

Симметрическая разность множеств (operator^=)

Временная сложность операции O(n).

Укорачивание (erase)

Временная сложность операции O(nlogn), т.к операция формирует результат одновременно со вставкой в дерево, причем вставка идет с поиском от корня, т.е за логарифмическое время.

Исключение (excl)

Временная сложность операции O(nlogn).

Включение (subst).

Временная сложность операции O(nlogn).

5. Пример работы программы

Пример для множеств мощности 10, для большей наглядности.

Исходные множества и последовательности:

```
ПРИМЕР РАБОТЫ С МНОЖЕСТВОМ:
МНОЖЕСТВО( A ) : 3 5 7 11 13 18 19 24 25 28
ПОСЛЕДОВАТЕЛЬНОСТЬ ( A ): < 13 18 28 24 3 5 19 25 7 11 >

МНОЖЕСТВО( B ) : 9 10 13 17 18 19 21 23 26 28
ПОСЛЕДОВАТЕЛЬНОСТЬ ( B ): < 21 19 9 23 17 18 13 26 10 28 >

МНОЖЕСТВО( C ) : 1 3 4 7 10 14 18 20 22 26
ПОСЛЕДОВАТЕЛЬНОСТЬ ( C ): < 14 1 18 10 20 3 7 22 26 4 >

МНОЖЕСТВО( D ) : 2 3 4 7 13 14 15 20 23 29
ПОСЛЕДОВАТЕЛЬНОСТЬ ( D ): < 20 4 2 3 14 23 13 15 7 29 >

МНОЖЕСТВО( E ) : 2 7 8 9 14 15 17 18 25 27
ПОСЛЕДОВАТЕЛЬНОСТЬ ( E ): < 2 7 8 14 27 25 15 9 17 18 >
```

Пример выполнения $(A \setminus B) \cup C \oplus D \cap E$:

```
ВЫЧИСЛЕНИЕ (a - b) | c ^ d & e
1-2-mycont (n=15 h=3) ----------------------------------> 11.....................................................
.....................................................................................................
.........................4.....................................................22.....
.....................................................................................................
.............2.....................7.................................18.....................25.....
.....................................................................................................
...........1.........3.............5.........10.....................15........20.............24........26.............
.....................................................................................................
.....................................................................................................
.....................................................................................................
.....................................................................................................
.....................................................................................................
.....................................................................................................
.....................................................................................................
.....................................................................................................
.....................................................................................................
МНОЖЕСТВО( F ) : 1 2 3 4 5 7 10 11 15 18 20 22 24 25 26
ПОСЛЕДОВАТЕЛЬНОСТЬ ( F ): < 1 2 3 4 5 7 10 11 15 18 20 22 24 25 26 >
```

Пример выполнения операции над последовательностью сцепление. Последовательность a присоединяется к последовательности b.

```
ПРИМЕР РАБОТЫ С ПОСЛЕДОВАТЕЛЬНОСТЬЮ:
МНОЖЕСТВО( A ) : 3 5 7 11 13 18 19 24 25 28
ПОСЛЕДОВАТЕЛЬНОСТЬ ( A ): < 13 18 28 24 3 5 19 25 7 11 >

МНОЖЕСТВО( B ) : 9 10 13 17 18 19 21 23 26 28
ПОСЛЕДОВАТЕЛЬНОСТЬ ( B ): < 21 19 9 23 17 18 13 26 10 28 >

СЦЕПЛЕНИЕ a.concat(b)
МНОЖЕСТВО( A ) : 3 5 7 9 10 11 13 17 18 19 21 23 24 25 26 28
ПОСЛЕДОВАТЕЛЬНОСТЬ ( A ): < 13 18 28 24 3 5 19 25 7 11 21 19 9 23 17 18 13 26 10 28 >
```

Пример выполнения операции над последовательностью слияние. Последовательность c сливается с последовательностью d.

```
МНОЖЕСТВО( C ) : 1 3 4 7 10 14 18 20 22 26
ПОСЛЕДОВАТЕЛЬНОСТЬ ( C ): < 14 1 18 10 20 3 7 22 26 4 >

МНОЖЕСТВО( D ) : 2 3 4 7 13 14 15 20 23 29
ПОСЛЕДОВАТЕЛЬНОСТЬ ( D ): < 20 4 2 3 14 23 13 15 7 29 >

СЛИЯНИЕ c.merge(d)
МНОЖЕСТВО( C ) : 1 2 3 4 7 7 10 13 14 15 18 20 22 23 26 29
ПОСЛЕДОВАТЕЛЬНОСТЬ ( C ): < 14 1 18 10 20 3 7 20 4 2 3 14 22 23 13 15 7 26 4 29 >
```

Пример выполения операции над последовательностью укорачивание. Из последовательности e удалим элементы с 1 по 3.

```
МНОЖЕСТВО( E ) : 2 7 8 9 14 15 17 18 25 27
ПОСЛЕДОВАТЕЛЬНОСТЬ ( E ): < 2 7 8 14 27 25 15 9 17 18 >

УКОРАЧИВАНИЕ e.erase(1, 3)
МНОЖЕСТВО( E ) : 2 9 15 17 18 25 27
ПОСЛЕДОВАТЕЛЬНОСТЬ ( E ): < 2 27 25 15 9 17 18 >
```

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.

Приложение

screen.h — функции для работы с экраном

mycont.h — реализация контейнера множество+последовательность

main.cpp — демонстрационная программа