O. FrozenSet

Ограничение времени	2 секунды
Ограничение памяти	256.0 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

https://gitlab.com/ibr11/cpp_psami_base/-/tree/main/frozen_set

Пришлите архив с файлами frozen_set.h, universal_hash.h и, возможно, другими файлами реализации.

FrozenSet

Реализуйте шаблонный класс `FrozenSet`, который представляет собой структуру данных для хранения фиксированного набора ключей с линейным (в среднем) временем построения, линейными в худшем случае затратами по памяти и константным в худшем случае поиском элемента. Аналог в `Python`: https://docs.python.org/3/library/stdtypes.html#frozenset.

Несмотря на то, что тесты проверяют корректность решения только на множестве целых неотрицательных чисел не превосходящих 10^9, класс должен уметь работать с объектами произвольных типов (при условии передачи корректного класса хеш функций).

Детали реализации

Хеш-таблица

Предлагается использовать алгоритм FKS, который был рассмотрен на лекции. Класс должен быть параметризован типом ключа, хеш-функции и поддерживать следующий функционал:

- Конструктор по умолчанию. Создается пустое множество;
- Конструктор от пары итераторов (начало и конец сохраняемой последовательности) и генератора псевдослучайных чисел (см. ниже). Сохраняет элементы переданного промежутка, используя генератор для подбора хеш-функций. Итераторы удовлетворяют категории ForwardIterator, гарантируется, что последовательность не содержит одинаковых ключей;
- Методы `Size`, `Empty`, `Clear` с привычной семантикой;
- Метод `bool Find(key)`. Осуществляет поиск элемента во множестве.

При хранении элементов в хеш таблицах второго уровня важно отличать заполненные ячейки от незаполненных. Хранение специальных значений (например, -inf для чисел или пустых строк для `std::string`) обладает очевидным недостатком: невозможно распознать ситуацию, при которой специальное значение в действительности входит во множество хранимых ключей. Для решения этой проблемы предлагается хранить объекты `std::optional<T>`, которые дополнительно хранят состояние `has_value`, сообщающее о том было ли установлено значение данного объекта или нет (подробности в задаче Optional).

Хеш-функция

Дополнительно реализуйте класс `UniversalHash`, представляющий универсальное семейство линейных хеш-функций по некоторому простому модулю (разбиралось на лекции). Интерфейс класса:

- Конструктор от двух аргументов (коэффициенты линейной функции) со значениями по умолчанию 1 и 0 (для коэффициента перед x и свободного члена соответственно);
- Перегруженная операция функционального вызова от числа;
- Статическая константа `kMaxValue`
  - максимальное значение хеш-функции (на единицу меньше простого модуля);
- Статическая функция `GenerateHash(generator)`, принимающая генератор псевдослучайных чисел и возвращающая случайную хеш-функцию (объект со случайными коэффициентами).

Подробнее про генерацию случайных чисел в задаче Random.