МФТИ, ФИВТ, ИВТ
Алгоритмы и структуры данных, осень 2020
Лабораторная работа «Хеши»

1 Теоретические вопросы (2 балла)

1.1 Примеры (1 балл)

Назовём семейство хеш-функций равномерно распределённым, если при равномерном случайном выборе хеш-функции случайная величина, равная значению хеш-функции на фиксированном элементе x, распределена на пространстве значений равномерно. Далее, 2-независимым семейством хеш-функций называется семейство, в котором описанные выше случайные величины для любых двух различных фиксированных x и y независимы. Приведите пример семейства хеш-функций, которое:

a) равномерно распределено, но не является 2-независимым;
b) является 2-независимым, но не является равномерно распределённым.

1.2 Последовательности ДНК (1 балл)

Рассмотрим задачу хеширования последовательностей ДНК. Напоминаем, что ДНК являются длинными последовательностями из четырёх аминокислот, которые мы будем обозначать буквами A, T, G и C. Пусть хеш-функция h принимает на вход ДНК и в качестве хеша возвращает сумму номеров аминокислот по какому-то большому простому модулю, а затем складывает в хеш-таблицу с открытым ключом. Чем плоха такая реализация хеширования?

2 Хеш-таблицы (6 баллов)

2.1 Реализация (2 балла)

Реализуйте на C++ следующие хеш-таблицы:

a) Открытый ключ, линейное пробирование;
b) Открытый ключ, квадратичное пробирование;
c) Открытый ключ, двойное хеширование;
d) Хеширование цепочками с расширением в 2 раза при заполненности в 75% / 95%;
e) Cuckoo hashtable.

Ваш класс должен быть шаблонным и наследоваться от интерфейса IHashSet. Поскольку хеш-функции бывают разные, ваш класс должен содержать внутри себя экземпляр хеш-функции (или несколько экземпляров). Хеш-функция может быть экземпляром одного из следующих классов:

- Для строк
  1. std::hash
  2. md5
  3. sha256
  4. Хеш-функция из задачи 1.2
  5. Хеширование из алгоритма Рабина-Карпа
  6. murmur3
- Для чисел
  1. std::hash
  2. Tabulation hashing
  3. Функция типа $(a_{n-1}k^{n-1} + \ldots + a_1k + a_0) \pmod p$

4. md5
5. sha256
6. murmur3

Для подсчёта хеш-функций, не освещённых на лекциях, разрешается использование сторонних библиотек.

2.2 Бенчмарки (3 балла)

Проведите эксперименты с разными комбинациями хеш-таблиц и хеш-функций и замерьте среднее время ответа на случайные запросы следующим образом:

- Переберите n от 10 до $100,000,000$ с увеличением выборки в 1.5 раза (то есть 10, $10 \cdot 1.5$, $10 \cdot 1.5^2$, ...)
- Для каждого n сгенерируйте 10 разных наборов ключей и на каждом из них проведите следующее
  - Сгенерируйте n ключей, половина из которых есть в исходных n, а другая — нет. Назовём этот набор Q. Не забудьте случайным образом перемешать Q.
  - **Начните замер времени работы, начиная с этого шага.**
  - Добавьте все ключи в таблицу.
  - По очереди переберите все ключи из Q и с вероятностью $\frac{1}{10}$ вызовите `remove` для этого ключа, в остальных случаях вызовите `find`.
  - **Остановите замер времени на текущей итерации.**
- Усредните предыдущий этап по всем 10 выборкам и поделите на $|Q|$, получив амортизированное время работы одной операции.
- Постройте график зависимости усреднённого амортизированного времени на одну операцию для разных сочетаний хеш-таблиц и хеш-функций. Отразите на графике дополнительно функции $\sqrt{n}$ и $\log n$.

2.3 Взлом 2-независимого семейства (1 балл)

Подберите такой нетривиальный набор запросов, чтобы амортизированное время работы для хеш-таблицы с открытым ключом и функции типа $(a_{N-1}k^{N-1} + \ldots + a_1 k + a_0) \pmod p$ для $N = 2$ было

- **(1 балл)** $\Omega(\log n)$, где n — общее число ключей в таблице;
- **(Бонус, 2 балла, предыдущий пункт не учитывается)** $\Omega(\sqrt{n})$.

3 Фильтры (5 баллов)

3.1 Основная часть (5 баллов)

Реализуйте на C++ класс Блум-фильтра и фильтра кукушки, которые хранят в себе необходимое количество хеш-функций, обладают точностью 99% и занимают не более 4Mb памяти (или больше — в зависимости от размера $L3$ кэша на вашей машине). Вычислите остальные параметры, исходя из этих, и приведите эти вычисления. При расчётах помните, что хеш-функции также занимают место в памяти. Аналогично предыдущей части подберите разные хеш-функции к разным таблицам и проверьте, что ваша реализация действительно помещается в 4Mb и имеет точность 99%. Для проверки точности выполняйте случайные запросы, проверяя их в дублирующем ваш фильтр `std::unordered_set`. После этого аналогично предыдущей части протестируйте все комбинации фильтр—хеш-функция на входных данных разных размеров, замерьте время работы и постройте графики.

3.2 Бонус (3 балла)

Прочитайте статью про XOR-filter и напишите небольшой отчёт о том, как устроена эта структура и какие основные идеи в ней применяются. Реализуйте его в дополнение к остальным двум фильтрам. После этого сравните время его работы с временем работы двух других фильтров на одинаковых данных.

4 HyperLogLog (бонус, 5 баллов)

- Реализуйте структуру данных HyperLogLog, использующую разные хеш-функции из списка выше.
- Поэкспериментируйте с хеш-функцией и длиной префикса хеша и добейтесь того, что ответы вашей структуры отличаются от правильных не больше чем на 10%.
- Тестируйте на данных порядка 10^9 чисел, проверяя точность ответа с помощью структуры данных `std::unordered_set`.

Указания по выполнению заданий

- В качестве выполненной работы нужно прислать архив, содержащий все исходники, которые вы использовали, и некоторый отчёт с графиками и комментариями по ходу выполнения работы.
- Обязательно комментируйте все графики, которые вы наблюдаете, делайте выводы. Не стесняйтесь делиться своими мыслями и переживаниями.
- На всех графиках обязательно должны быть **подписаны оси**, адекватно **проставлены значимые отметки на осях** и обязательно должна присутствовать **легенда**.
- Как правильно работать со случайными числами в современном C++.
- Чтобы замерять время работы программы, используйте библиотеку std::chrono.
- Отчёт рекомендуется собирать в Jupyter notebook'е, создав .pdf с кодом и графиками. Графики рекомендуется рисовать с помощью библиотеки matplotlib.pyplot языка Python.