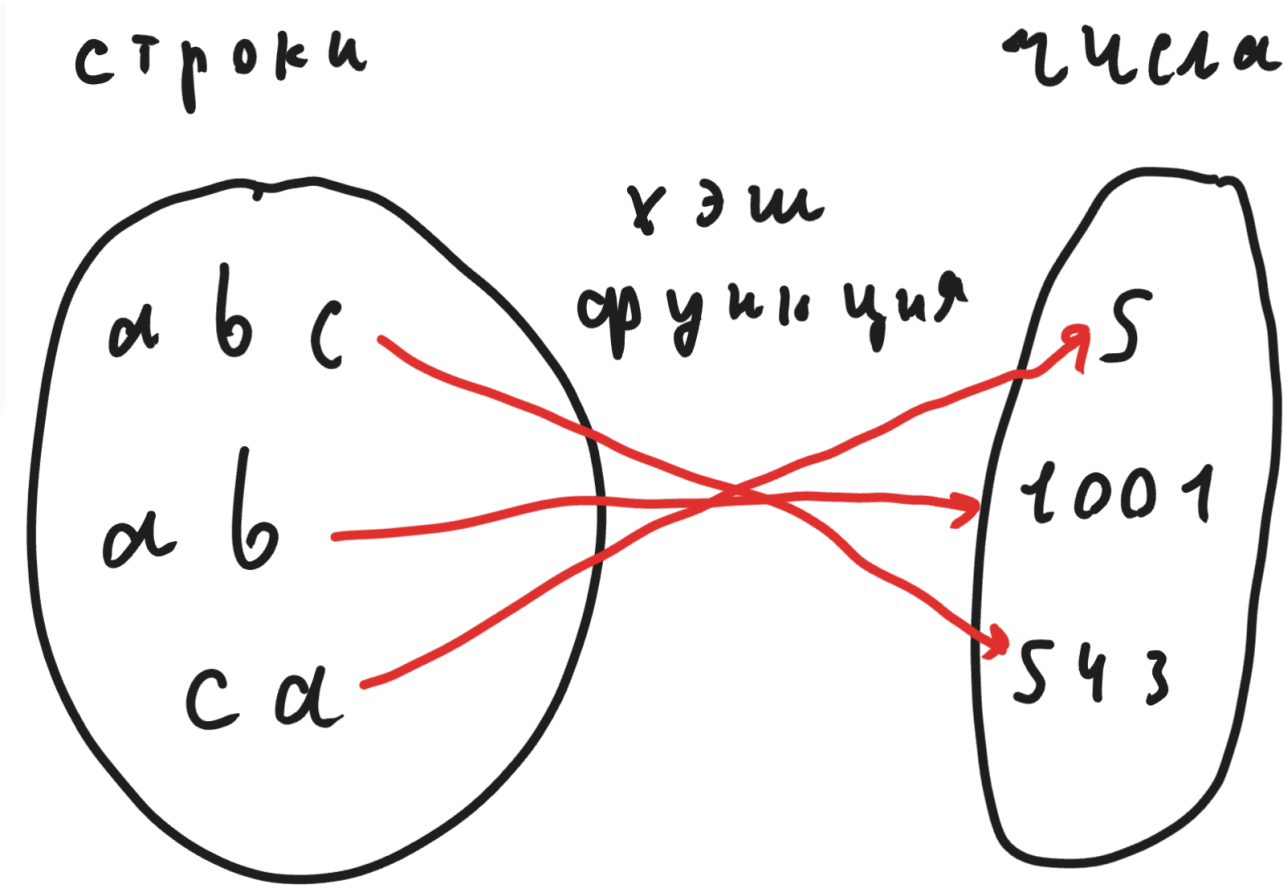
Хеширование

Хеши

Часто возникает потребность проверять равенство двух объектов.
Это делать легко, когда объекты - числа. Но что делать, если объекты - строки, графы и другие "большие" объекты? В таком случае проверка равенства может быть долгой.

Давайте будем превращать эти большие объекты, которые проверять на равенство сложно в числа, которые проверять легко. Функция "превращения" будет называться хеш функцией $F(x)$



Требования к хеш-функции

Хеш функции можно поделить на два класса: криптографические и общего назначения. Требования к криптографическим функциям другие.

Требования к хеш функциям общего назначения:

- **Детерминированность**. Хеш функция для одного и того же объекта должна давать один и тот же хеш независимо от запуска
- **Быстрота**. Должна считаться за O(1) и очень быстро
- **Равномерное распределение**. Хеши должны равномерно "размазываться".

Ситуативные требования (выдвигаются в зависимости от задачи):

- Preimage resistance - устойчивость к поиску прообраза. Зная F(m) должно быть крайне сложно восстановить m
- Second preimage resistance - зная m1 и F(m1) должно быть сложно найти m2 $\neq$ m1 с таким же хешем.
- Collision resistance - крайне сложно найти любые два разных сообщения $m1 \neq m2$, $F(m1) = F(m2)$
- Лавинный эффект - изменение 1 бита входа -> ~50% битов хеша меняется.
- Псевдослучайность - выход выглядит как случайные биты.

Где используется:

- Проверка паролей (не храним исходный пароль, а только хеш)
- Хеш таблица (ассоциативный массив)
- Криптографические хеш-функции (цифровые подписи, проверка целостности файлов и т.д.)
- Блокчейн
- Базы данных (хеш-индексы)
- DHT (Distributed Hash Tables) - BitTorrent, Kademlia
- Git - каждый коммит, дерево, блоб - это SHA-1
- Торрент файлы

Основная проблема хеширования

Различных строк очень много, в сравнении с ними, чисел сильно меньше. Ситуация, когда $m1 \neq m2, F(m1) = F(m2)$. Объекты разные, а хеши совпадают.

Задача 1.

Пусть даны два **неупорядоченных** множества различных чисел (произвольных, не случайных). Нужно определять, равны множества или нет

Наивное решение

Отсортируем множества (упорядочим), сравним с помощью одного цикла, что все элементы на одинаковых позициях совпадают

Асимптотика

Сортировка $O(n \log n)$ + проход по списку $O(n)$

Быстрое решение

Сопоставим каждому числу из множеств какое-то случайное число (хеш)
Пусть хеш множества будет высчитываться как xor всех хешей элементов.

Если считать, что хеши равномерно распределены, то каждый хеш равновероятен.
Если каждый хеш равновероятен, то рассмотрим вероятность появления каждого бита. Вероятность каждого отдельно взятого бита $0,5$
xor двух равновероятных битов - равновероятен
(можно в этом убедиться, если руками посчитать вероятности)
Следовательно, хеш списка равновероятен. Вероятность совпадения хешей при несовпадающих массивах крайне мала из-за взятия случайных чисел в качестве хеша.

Асимптотика

Хеш функция $O(1)$
Сопоставление - hash table $O(1)$
Подсчёт и проверка $O(n)$

Задача 2.

Допустим, имеется строка S, $|S| = n$.
Строка состоит из латинских символов нижнего регистра.
Поступает q запросов вида "совпадают ли $S[l1...r1]$ $и$ $S[l2...r2]$"

Наивное решение

Честно сравнивать подстроки на совпадение. $O(n)$ на каждый запрос, $O(n\ q)$ суммарно

Быстрое решение

Научимся считать хеш так, чтобы брать хеш подстроки за $O(1)$

Полиномиальный хеш

Возьмём $m = 10^9 + 7$ и $p = 31$. Посчитаем хеш каждого префикса строки S

```
P[i] = S[0] * p^0 + S[1] * p^1 + ... + S[i] * p^i mod m
```

$P[i] = \sum_{j=0}^{i} s[j] * p^j \ mod \ m$

Воспользуемся идеей префикс сумм
$P[r] - P[l-1] = \sum_{i=0}^{r} s[i] * p^i - \sum_{j=0}^{l-1} s[j] * p^j = \sum_{i=l}^{r} s[i] * p^i = s[l] * p^l + s[l+1] * p^{l+1} +$.
Это и будет полиномиальной хеш функцией. Строим полином для основания p (представляем строку в системе счисления с основанием p по модулю m)

Сравнение двух подстрок

$P[r1] - P[l1-1] = s[l1] * p^{l1} + s[l1+1] * p^{l1+1} + \ldots + s[r1] * p^{r1} \ mod \ m$
$P[r2] - P[l2-1] = s[l2] * p^{l2} + s[l2+1] * p^{l2+1} + \ldots + s[r2] * p^{r2} \ mod \ m$
Если символы $s[i]$ будут совпадать, то хеши будут отличаться из-за разных степеней

Решение 1.

Поделить оба значения на p^{l1} и p^{l2} соответственно (неприятно делить в кольце по модулю)

Решение 2.

Домножить оба значения на p^{n-l1} и p^{n-l2} соответственно

Итоговая асимптотика

Предподсчёт хешей, степеней по модулю $O(n)$
Сравнение подстрок $O(1)$
Итого: $O(n + q)$

Задача 3.

Поиск наибольшего общего префикса

Решение

Если совпадает префикс длины l, то совпадает и префикс длины l - 1.
Можно запустить бинарный поиск по длине совпадающего префикса.
Проверять совпадает ли префикс можно хешами за $O(1)$
Итого: $O(n\ log\ n)$

Задача 4.

Поиск вхождений строки T в строке S

Решение

Посчитаем полиномиальный хеш строки T
Пройдём очередью длины $|T|$ по строке S и будем поддерживать хеш текущей подстроки.

Коллизия и способы борьбы с ней

Вероятность коллизии можно посчитать используя теорию вероятностей и предполагая, что каждый хеш равновероятен.

- Взять простой модуль порядка 10^{18}

- Добавить второй хеш (третий, четвёртый) с другими параметрами (модуль, основание)