Theory

Claim Утверждение 1

В данной хэш функции используется знаковая арифметика, тип **long long**
Считая, что используется беззнаковая арифметика, то есть тип **unsigned long long**, результат работы хэш функции не изменится (то есть по-битово значения будут совпадать)

Proof:

В хэш функции используются только операции умножения и сложения
Тогда результат их работы не отличается для знаковых и беззнаковых типов
ввиду реализации сложения и умножения в АЛУ (см. инструкции **add** и **mul**)

■

Clarification Уточнение

Далее в рассуждениях используются следующие обозначения:

- $\Sigma = \{'a', 'b', ..., 'z'\}$ - символы таблицы ASCII с кодами от 97 до 122
- $\sigma = \{1, 2, ..., 26\}$
  При дальнейшем описании и построении искомого алгоритма будем считать,
  что используются только символы таблицы ASCII из множества Σ
- Σ^* - замыкание Клини, т.е. множество всех слов конечной длины,
  составленных из символов множества Σ, в том числе пустое слово ε, $|\varepsilon| = 0$
- Σ^+ - плюс Клини, т.е. множество всех слов конечной длины,
  составленных из символов множества Σ, без пустого слова ε
  $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$
- p - данное простое число, $p = 31$
- $\text{Hash}(s)$ - математическое значение полиномиальной хэш функции строки $s \in \Sigma^*$
  без операция взятия остатка по модулю 2^{64}
- $\text{hash}(s)$ - значение хэш функции строки $s \in \Sigma^*$ по модулю 2^{64}, т.е.
  $\forall s \in \Sigma^* : \text{hash}(s) = \text{Hash}(S) \bmod 2^{64}$
  По утверждению 1 можно считать, что данная полиномиальная хэш функция - hash
- $\overline{s_0 s_1 ... s_{n-1}}$ - строка, состоящая из последовательности строк $\{s_i\}_{i=0}^{n-1}$,
  последовательно сконкатенированных в одну строку
  (мы не определяем это формально, потому что ~~автору лень~~ тогда мы утонем в формализме
  при доказательстве задачи, которая решается за 15 минут~~, а ещё автору лень~~)

Lemma Лемма 1

При подсчёте хэш функции Hash от строки $s \in \Sigma^*$ получается число в системе счисления по основанию p, где цифры числа $\text{Hash}(s)$ в с.с. по основанию p соответствуют символам строки s и принадлежат множеству σ

Proof:

Следует из того, что при подсчёте хэш функции $\forall i \in \{0, 1, ..., |s| - 1\} : s[i] -' a' + 1 \in \sigma \subset \{0, 1, ..., p - 1\}$

И при этом $s[i] -' a' + 1$ умножается на p^i : $\text{Hash}(s) = \sum_{i=0}^{|s|-1} (s[i] -' a' + 1) \cdot p^i$

■

Lemma Лемма 2

Пусть существует такая строка $s_0 \in \Sigma^+$ длины $|s_0|$, что $\text{hash}(s_0) = 0$
Тогда $\forall n \in \mathbb{N} : \text{hash}(\underbrace{\overline{s_0 s_0 ... s_0}}_{n \text{ раз}}) = 0$

Proof:

1. По построению хэш функции Hash:

$$\text{Hash}(s_0) = \sum_{i=0}^{|s_0|-1} (s_0[i] -' a' + 1) \cdot p^i$$

2. Тогда:

$$\text{Hash}(\overline{s_0 s_0 ... s_0}) = \sum_{i=0}^{n \cdot |s_0|-1} (\overline{s_0 s_0 ... s_0}[i] +' a' - 1) \cdot p^i =$$

$$= \sum_{j=0}^{n-1} \sum_{i=|s_0| \cdot j}^{|s_0| \cdot (j+1)-1} (\overline{s_0 s_0 ... s_0}[i] +' a' - 1) \cdot p^i =$$

$$= \sum_{j=0}^{n-1} p^{|s_0| \cdot j} \sum_{i=|s_0| \cdot j}^{|s_0| \cdot (j+1)-1} (\overline{s_0 s_0 ... s_0}[i] +' a' - 1) \cdot p^{i-|s_0| \cdot j} =$$

$$= \sum_{j=0}^{n-1} p^{|s_0| \cdot j} \sum_{i=0}^{|s_0|-1} (\overline{s_0 s_0 ... s_0}[i + |s_0| \cdot j] +' a' - 1) \cdot p^i =$$

$$= \sum_{j=0}^{n-1} p^{|s_0| \cdot j} \sum_{i=0}^{|s_0|-1} (s_0[i] +' a' - 1) \cdot p^i = \sum_{j=0}^{n-1} p^{|s_0| \cdot j} \cdot \text{Hash}(s_0) = \text{Hash}(s_0) \cdot \sum_{j=0}^{n-1} p^{|s_0| \cdot j}$$

$$\text{hash}(s_0) = 0 \implies \text{Hash}(s_0) \bmod 2^{64} = 0 \implies \text{hash}(\overline{s_0 s_0 ... s_0}) = \text{Hash}(\overline{s_0 s_0 ... s_0}) \bmod 2^{64} =$$

$$= \left(\text{Hash}(s_0) \cdot \sum_{j=0}^{n-1} p^{|s_0| \cdot j} \right) \bmod 2^{64} = 0$$

■

Lemma Лемма 3

Пусть существует такая строка $s_0 \in \Sigma^+$ длины $|s_0|$, что $\text{hash}(s_0) = 0$
Тогда $\forall s \in \Sigma^* \, \forall n \in \mathbb{N} : \text{hash}(\overline{\underbrace{ss_0s_0...s_0}_{n \text{ раз}}}) = \text{hash}(s)$

Proof:

1. По построению хэш функции:

$$\text{Hash}(\overline{ss_0s_0...s_0}) = \sum_{i=0}^{|s|+n\cdot|s_0|-1} (\overline{ss_0s_0...s_0}[i] -' a' + 1) \cdot p^i =$$

$$= \sum_{i=0}^{|s|-1} (\overline{ss_0s_0...s_0}[i] +' a' - 1) \cdot p^i + \sum_{i=|s|}^{|s|+n\cdot|s_0|-1} (\overline{ss_0s_0...s_0}[i] +' a' - 1) \cdot p^i$$

$$= \sum_{i=0}^{|s|-1} (s[i] +' a' - 1) \cdot p^i + p^{|s|} \cdot \sum_{i=|s|}^{|s|+n\cdot|s_0|-1} (\overline{ss_0s_0...s_0}[i] +' a' - 1) \cdot p^{i-|s|} =$$

$$= \text{Hash}(s) + p^{|s|} \cdot \sum_{i=|s|}^{|s|+n\cdot|s_0|-1} (\overline{ss_0s_0...s_0}[i] +' a' - 1) \cdot p^{i-|s|} =$$

$$= \text{Hash}(s) + p^{|s|} \cdot \sum_{i=0}^{n\cdot|s_0|-1} (\overline{ss_0s_0...s_0}[i + |s|] +' a' - 1) \cdot p^i =$$

$$= \text{Hash}(s) + p^{|s|} \cdot \sum_{i=0}^{n\cdot|s_0|-1} (\overline{s_0s_0...s_0}[i] +' a' - 1) \cdot p^i =$$

$$= \text{Hash}(s) + p^{|s|} \cdot \text{Hash}(\overline{s_0s_0...s_0})$$

2. по лемме 2 $\text{hash}(\overline{s_0s_0...s_0}) = 0$, тогда:

$$\text{hash}(\overline{ss_0s_0...s_0}) = \text{Hash}(\overline{ss_0s_0...s_0}) \bmod 2^{64} = \text{Hash}(s) \bmod 2^{64} + \left(p^{|s|} \cdot \text{Hash}(\overline{s_0s_0...s_0})\right) \bmod 2^{64} =$$

$$= \text{hash}(s) + \left(p^{|s|} \cdot \text{hash}(\overline{s_0s_0...s_0})\right) \bmod 2^{64} = \text{hash}(s)$$

■

Corollary Следствие

Если существует хотя бы одна строка $s_0 \in \Sigma^*$, такая что $|s_0| \geq 1 \wedge \text{hash}(s_0) = 0$, то $\forall s \in \Sigma^*$ можно получить хотя бы счётно бесконечно много попарно различных строк, таких что их хэш равен $\text{hash}(s)$, причём алгоритм построения искомой последовательности строк следует из леммы 3: члены последовательности получаются конкатенацией строки s с n строками s_0.

Clarification Уточнение об имплементации алгоритма

Если длина выбранной строки s_0 (для которой $\text{hash}(s_0) = 0$) равна $|s_0|$ и нужно сгенерировать N искомых строк с хэшом, равным данной строке s длины $|s|$, то асимптотически точная граница времени работы приведённого ниже алгоритма равна $\Theta(|s| + N|s_0|)$
В приведённом ниже алгоритме строка s_0 (можно генерировать разные в зависимости от шаблонного параметра) имеет длину не более $kMaxZeroRemStringSize = 20$ и генерируется на этапе компиляции (если версия языка $\geq$ C++23).

Practice

Алгоритм реализован в файле *collisions_gen.hpp*
Публичный интерфейс для генерации строк находится в пространстве имён collisions_gen
Реализация находится во вложенном пространстве имён collisions_gen::impl
Публичный интерфейс:

- *polynomial_hash* - данная хэш функция
- *polynomial_hash_safe* - данная хэш функция, использующая беззнаковую арифметику
- *generate_strings_with_same_hash* - функция для генерации
  необходимого количества строк, хэш которых равен хэшу данной строки
  Кроме 2 входных аргументов имеет шаблонный параметра Seed, который
  может изменить генерирующиеся строки. По умолчанию равен *impl::kDefaultStartN = 1*
  следующее по возрастанию значение, меняющее выходные данные = 13

```
1
2  namespace collisions_gen {
3
4  /// @brief Default number of strings generated by @fn generate_strings_with_same_hash
5  inline constexpr uint32_t kDefaultSize = 2000;
6
7  /// @brief Polynomial hash function.
8  ///        This function causes UB and hence not marked constexpr.
9  /// @param s
10 /// @return polynomial hash of the @a `s`
11 long long polynomial_hash(std::string_view s) noexcept;
12
13 /// @brief Polynomial hash function.
14 /// @param s
15 /// @return polynomial hash of the @a `s`
16 constexpr uint64_t polynomial_hash_safe(std::string_view s) noexcept;
17
18 /// @brief Functions that generates @a `size` strings with the same
19 ///        hash as @a `str` has. Hash functions is @fn polynomial_hash.
20 /// @tparam Seed optional argument that may change generated set of string.
21 /// @param str Initial string.
22 ///            If empty, and Seed = @ref impl::kDefaultStartN,
23 ///            then @a `size` strings with hash = 0 will be generated.
24 /// @param size Number of strings that should be generated.
25 /// @return vector of @a `size` pairwise different strings,
26 ///         each one has the same hash as @a `str` has.
27 template <uint32_t Seed = impl::kDefaultStartN>
28 std::vector<std::string> generate_strings_with_same_hash(std::string_view str = "",
29 uint32_t size = kDefaultSize);
30
31 }  // namespace collisions_gen
```

Вызов генератора строк находится в файле *main.cpp*

В качестве первого аргумента исполняемому файлу можно передать начальную строку, к которой будут дописываться строки s_0

В качестве второго аргумента исполняемому файлу можно передать количество строк, которое надо сгенерировать

(также можно не передавать никаких параметров, будут выбраны дефолтные: пустая строка и 2000 строк)

```
1
2  void write_to_file(const std::vector<std::string>& strs, std::string_view fname);
3
4  std::pair<const char*, uint32_t> parse_arguments(int argc, const char* const argv[]) noexcept;
5
6  int main(int argc, const char* const argv[]) {
7      auto [initial_string, size] = parse_arguments(argc, argv);
8      auto res = collisions_gen::generate_strings_with_same_hash(initial_string, size);
9      write_to_file(res, "strings.txt");
10     return 0;
11 }
```

Tested compilers & options

При компиляции компилятором g++ 13.2.0 из среды msys2 на Windows 10 22H2 использовались следующие флаги:

-std=c++20, -std=c++2a, -std=c++2b для версий языка C++20 и C++23, а также:

```
-D_GLIBCXX_DEBUG
-D_GLIBCXX_DEBUG_PEDANTIC
-D_FORTIFY_SOURCE=3
-fdiagnostics-color=always
-fstack-protector-all
-mshstk
-Wall
-Wextra
-Wfloat-equal
-Wlogical-op
-Wcast-qual
-Wpedantic
-Wshift-overflow=2
-Wduplicated-cond
-Wunused -Wconversion
-Wunsafe-loop-optimizations
-Wshadow
-Wnull-dereference
-Wundef
-Wwrite-strings
-Wsign-conversion
-Warith-conversion
-Wmissing-noreturn
-Wunreachable-code
-Wcast-align
-Warray-bounds=2
```

При компиляции компилятором clang++ 16.0.5 из среды msys2 на Windows 10 22H2 использовались следующие флаги:

-std=c++20, -std=c++2a, -std=c++2b для версий языка C++20 и C++23, а также:

```
-fcolor-diagnostics
-fansi-escape-codes
-fsanitize="address,undefined"
-fstack-protector-all
-D_FORTIFY_SOURCE=3
-Wp,-D_GLIBCXX_DEBUG
-mshstk
-O2
-Wall
-Wextra
-Wpedantic
-Wunused
-Wconversion
-Wshadow
-Wnull-dereference
-Wundef
-Wwrite-strings
-Wsign-conversion
-Wmissing-noreturn
-Wunreachable-code
-Wcast-align
-Warray-bounds
```