

## 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:

- $\Sigma = \{ 'a', 'b', \dots, 'z' \}$  - символы таблицы ASCII с кодами от 97 до 122
- $\sigma = \{ 1, 2, \dots, 26 \}$

При дальнейшем описании и построении искомого алгоритма будем считать, что используются только символы таблицы ASCII из множества  $\Sigma$

- $\Sigma^*$  - замыкание Клини, т.е. множество всех слов конечной длины, составленных из символов множества  $\Sigma$ , в том числе пустое слово  $\epsilon$ ,  $|\epsilon| = 0$
- $\Sigma^+$  - плюс Клини, т.е. множество всех слов конечной длины, составленных из символов множества  $\Sigma$ , без пустого слова  $\epsilon$   
 $\Sigma^+ = \Sigma^* \setminus \{ \epsilon \}$
- $p$  - 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 -  $\text{hash}$

- $s_0 s_1 \dots s_{n-1}$  - 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  $\text{Hash}$  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  $\sigma$

#### *Proof:*

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

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  $\forall n \in \mathbb{N} : \text{hash}(\overbrace{s_0 s_0 \dots s_0}^n \text{ раз}) = 0$

**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:

$$\begin{aligned} \text{Hash}(\overbrace{s_0 s_0 \dots s_0}^n) &= \sum_{i=0}^{n \cdot |s_0| - 1} (\overbrace{s_0 s_0 \dots s_0}^n[i] - 'a' + 1) \cdot p^i = \\ &= \sum_{j=0}^{n-1} \sum_{i=|s_0| \cdot j}^{|s_0| \cdot (j+1) - 1} (\overbrace{s_0 s_0 \dots s_0}^n[i] - 'a' + 1) \cdot p^i = \\ &= \sum_{j=0}^{n-1} p^{|s_0| \cdot j} \sum_{i=|s_0| \cdot j}^{|s_0| \cdot (j+1) - 1} (\overbrace{s_0 s_0 \dots s_0}^n[i] - 'a' + 1) \cdot p^{i - |s_0| \cdot j} = \\ &= \sum_{j=0}^{n-1} p^{|s_0| \cdot j} \sum_{i=0}^{|s_0| - 1} (\overbrace{s_0 s_0 \dots s_0}^n[i + |s_0| \cdot j] - 'a' + 1) \cdot p^i = \\ &= \sum_{j=0}^{n-1} p^{|s_0| \cdot j} \sum_{i=0}^{|s_0| - 1} (s_0[i] - 'a' + 1) \cdot p^i = \sum_{j=0}^{n-1} p^{|s_0| \cdot j} \cdot \text{Hash}(s_0) = \text{Hash}(s_0) \cdot \sum_{j=0}^{n-1} p^{|s_0| \cdot j} \\ \text{hash}(s_0) = 0 &\implies \text{Hash}(s_0) \bmod 2^{64} = 0 \implies \text{hash}(\overbrace{s_0 s_0 \dots s_0}^n) = \text{Hash}(\overbrace{s_0 s_0 \dots s_0}^n) \bmod 2^{64} = \\ &= \left( \text{Hash}(s_0) \cdot \sum_{j=0}^{n-1} p^{|s_0| \cdot j} \right) \bmod 2^{64} = 0 \end{aligned}$$

■

### Lemma Лемма 3

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

**Proof:**

1. По построению хэш функции:

$$\begin{aligned} \text{Hash}(\overbrace{ss_0s_0\dots s_0}^n) &= \sum_{i=0}^{|s|+n\cdot|s_0|-1} (\overbrace{ss_0s_0\dots s_0}^n[i] - 'a' + 1) \cdot p^i = \\ &= \sum_{i=0}^{|s|-1} (\overbrace{ss_0s_0\dots s_0}^n[i] + 'a' - 1) \cdot p^i + \sum_{i=|s|}^{|s|+n\cdot|s_0|-1} (\overbrace{ss_0s_0\dots s_0}^n[i] + 'a' - 1) \cdot p^i \\ &= \sum_{i=0}^{|s|-1} (s[i] + 'a' - 1) \cdot p^i + p^{|s|} \cdot \sum_{i=|s|}^{|s|+n\cdot|s_0|-1} (\overbrace{ss_0s_0\dots s_0}^n[i] + 'a' - 1) \cdot p^{i-|s|} = \\ &= \text{Hash}(s) + p^{|s|} \cdot \sum_{i=|s|}^{|s|+n\cdot|s_0|-1} (\overbrace{ss_0s_0\dots s_0}^n[i] + 'a' - 1) \cdot p^{i-|s|} = \\ &= \text{Hash}(s) + p^{|s|} \cdot \sum_{i=0}^{n\cdot|s_0|-1} (\overbrace{ss_0s_0\dots s_0}^n[i + |s|] + 'a' - 1) \cdot p^i = \\ &= \text{Hash}(s) + p^{|s|} \cdot \sum_{i=0}^{n\cdot|s_0|-1} (\overbrace{s_0s_0\dots s_0}^n[i] + 'a' - 1) \cdot p^i = \\ &= \text{Hash}(s) + p^{|s|} \cdot \text{Hash}(\overbrace{s_0s_0\dots s_0}^n) \end{aligned}$$

2. по лемме 2  $\text{hash}(\overbrace{s_0s_0\dots s_0}^n) = 0$ , тогда:

$$\begin{aligned} \text{hash}(\overbrace{ss_0s_0\dots s_0}^n) &= \text{Hash}(\overbrace{ss_0s_0\dots s_0}^n) \bmod 2^{64} = \text{Hash}(s) \bmod 2^{64} + (p^{|s|} \cdot \text{Hash}(\overbrace{s_0s_0\dots s_0}^n)) \bmod 2^{64} = \\ &= \text{hash}(s) + (p^{|s|} \cdot \text{hash}(\overbrace{s_0s_0\dots s_0}^n)) \bmod 2^{64} = \text{hash}(s) \end{aligned}$$

■

### 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 хэшем, 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
```

Пространство имён *impl*:

(некоторые детали реализации, например, поддержка более старых версий языка, убраны в коде ниже, полный код есть в файле *collisions\_gen.hpp*)

- *find\_zero\_rem\_num* - функция нахождения такого  $p$ , что все цифры числа  $n \cdot 2^{64}$  в с.с. по основанию  $p$  принадлежат  $\sigma$
- *ZeroHashStringBuffer* - шаблонный класс, который по данному числу  $n$  переводит число  $n \cdot 2^{64}$  в с.с. по основанию  $p$  и сохраняет в виде строки
- *zero\_hash\_string* - шаблонная константа типа *std::string\_view*, такая, что её хэш равен 0

```
1
2 namespace impl {
3
4 using uint128_t = __uint128_t;
5
6 inline constexpr uint32_t kPrime = 31;
7 inline constexpr uint32_t kDefaultStartN = 1;
8 /// @brief @f[\lceil \log_{\{kPrime\}}(2^{\{32\}} \cdot 2^{\{64\}}) \rceil @f]
9 inline constexpr uint32_t kMaxZeroRemStringSize = 20;
10
11 /// @brief f: { 0, ..., 2^{32} - 1 } -> { 0, 1 }
12 /// @param n
```

```

13 /// @return n |-> n \in [1; 26]
14 constexpr bool between_1_and_26(uint32_t n) noexcept;
15
16 /// @brief Checks whether all digits of n
17 ///         in base @ref kPrime \in [1; 26]
18 /// @param n
19 /// @return
20 constexpr bool check_n_in_base_p(uint128_t n) noexcept;
21
22 /// @brief Finds integer n, such that all digits
23 ///         of  $n * 2^{64}$  in base @ref kPrime \in [1; 26]
24 /// @tparam StartN
25 /// @return n
26 template <uint32_t StartN = kDefaultStartN>
27 constexpr uint32_t find_zero_rem_num() noexcept;
28
29 template <uint32_t N>
30 struct ZeroHashStringBuffer {
31     consteval ZeroHashStringBuffer() noexcept;
32     consteval std::string_view as_string_view() const noexcept;
33 };
34
35 template <uint32_t StartN = kDefaultStartN>
36 inline constexpr ZeroHashStringBuffer<find_zero_rem_num<StartN>()> zero_hash_string_buffer;
37
38 template <uint32_t StartN = kDefaultStartN>
39 inline constexpr std::string_view zero_hash_string
40     = zero_hash_string_buffer<StartN>.as_string_view();
41
42 } // namespace impl

```

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>& str, 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:

```
1 -D_GLIBCXX_DEBUG
2 -D_GLIBCXX_DEBUG_PEDANTIC
3 -D_FORTIFY_SOURCE=3
4 -fdiagnostics-color=always
5 -fstack-protector-all
6 -mshstk
7 -Wall
8 -Wextra
9 -Wfloat-equal
10 -Wlogical-op
11 -Wcast-qual
12 -Wpedantic
13 -Wshift-overflow=2
14 -Wduplicated-cond
15 -Wunused -Wconversion
16 -Wunsafe-loop-optimizations
17 -Wshadow
18 -Wnull-dereference
19 -Wundef
20 -Wwrite-strings
21 -Wsign-conversion
22 -Warith-conversion
23 -Wmissing-noreturn
24 -Wunreachable-code
25 -Wcast-align
26 -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:

```
1 -fcolor-diagnostics
2 -fans-escape-codes
3 -fsanitize="address,undefined"
4 -fstack-protector-all
5 -D_FORTIFY_SOURCE=3
6 -Wp,-D_GLIBCXX_DEBUG
7 -mshstk
8 -O2
9 -Wall
10 -Wextra
11 -Wpedantic
12 -Wunused
13 -Wconversion
14 -Wshadow
15 -Wnull-dereference
16 -Wundef
17 -Wwrite-strings
18 -Wsign-conversion
19 -Wmissing-noreturn
20 -Wunreachable-code
21 -Wcast-align
22 -Warray-bounds
```

При компиляции компилятором g++ 13.2.0 на Ubuntu 22.10 использовались следующие флаги:

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

```
1 -D_GLIBCXX_DEBUG
2 -D_GLIBCXX_DEBUG_PEDANTIC
3 -D_FORTIFY_SOURCE=3
4 -fdiagnostics-color=always
5 -fstack-protector-all
6 -fsanitize="address,undefined,leak"
7 -mshstk
```

```
8 -Wall
9 -Wextra
10 -Wfloat-equal
11 -Wlogical-op
12 -Wcast-qual
13 -Wpedantic
14 -Wshift-overflow=2
15 -Wduplicated-cond
16 -Wunused -Wconversion
17 -Wunsafe-loop-optimizations
18 -Wshadow
19 -Wnull-dereference
20 -Wundef
21 -Wwrite-strings
22 -Wsign-conversion
23 -Warith-conversion
24 -Wmissing-noreturn
25 -Wunreachable-code
26 -Wcast-align
27 -Warray-bounds=2
```

---