SET-5

Группа БПИ243, Тупицин Тимофей Романович

Задача A2b. Взломщик!

Для хеширования строковых ключей, которые могут содержать строчные/прописные латинские буквы и цифры, используется следующая полиномиальная хеш-функция, значение которой определяется числовым параметром p:

```
size_t hash(std::string key) {
    const int p = ???;
    long long h = 0, p_pow = 1;
    for (size_t i = 0; i < s.length(); ++i) {
        h += (s[i] - 'a' + 1) * p_pow;
        p_pow *= p;
    }
    return h;
}
```

Для того, чтобы взломать хеш-функцию, требуется найти такой набор строк, который вызывает *коллизии* — одинаковые значения хеш-функции.

1. Алгоритм поиска нейтральных элементов.
Известно, как считаетс хэщ-функция для строки s из 2 элементов.

$$hash = (s[0] -' a' + 1) * p + (s[1] -' a' + 1) * p^2$$

Хэщ нейтрального элемента должен быть равен 0.

$$0 = (s[0] -' a' + 1) * p + (s[1] -' a' + 1) * p^2$$

вынесем p за скобки, получим

$$0 = s[0] -' a' + 1 + (s[1] -' a' + 1) * p$$

Скажем, что $s[0] -' a' + 1 = f(s[0])$, тогда

$$0 = f(s[0]) + f(s[1]) * p \Rightarrow f(s[0]) = -f(s[1]) * p$$

хэш первого элемента должен быть равен минус хэшу второго элемента, умноженнному на p.

Алгоритм будет работать следующим образом: мы будем перебирать все возможные символы для второго элемента строки, вычислять их хэш и умножать на p, а затем искать символ для первого элемента строки, который будет иметь хэш, равный необходимомму хэщу. Если такой символ существует, то мы нашли нейтральный элемент.

Сам алгоритм:

1. Сначала определяем наш алфавит. 52 символа (26 строчных и 26 заглавных букв) + 10 цифр.
2. для каждого p перебираем все символы для второго элемента строки $c1$.
3. для каждого символа $c1$ вычисляем необходимый хэш для первого элемента строки $c0$.
4. проверяем, существует ли символ $c0$ в нашем алфавите, который имеет хэш, равный необходимому хэшу. Если да, то мы нашли нейтральный элемент.

Обоснование

Условие $f(s[0]) + f(s[1]) * p = 0$ напрямую выведено из формулы хеша для двух символов. Если оно выполняется, хеш равен 0 по определению. Полный перебор всех 62 символов алфавита гарантирует, что мы найдем решение для любого p, если оно существует. Алгоритм работает быстро, так как количество возможных комбинаций символов очень мало.

main.cpp

```cpp
int main() {
    std::string str = "qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM1234567890";
    for (int p = 0; p <= 31; ++p) {
        std::cout << "p = " << p << ": ";
        for (char c1 : str) {
            int f1 = (int)c1 - 'a' + 1;
            int f0 = -p * f1;
            for (char c0 : str) {
                if ((int)c0 - 'a' + 1 == f0) {
                    std::cout << c0 << c1 << " ";
                }
            }
        }
        std::cout << "\n";
    }
    return 0;
}
```

Рис. 1: Реализация алгоритма на языке C++

2. Все нейтральные элементы для $p \leq 31$.

p = 0:

p = 1: Oq Iw Nr Lt Gy Ku Wi Qo Pp Ms Zf Yg Xh Vj Uk Tl Fz Hx Jv Rn Sm oQ iW nR lT gY kU wI qO pP mS zF yG xH vJ uK tL fZ hX jV rN sM

p = 2: 2w Ve 8t 6u Ni Bo Xd Tf Rg Ph Lj Jk Hl 0x Zc 4v Dn Fm rW xT nY vU zS lZ pX tV

p = 3: Qe Ei 3o 0p Td Nf Kg Hh Bj Wc Zb 6n 9m uY rZ xX

p = 4: Le Pd Hf Dg 8j 4k 0l Tc Xb xZ

p = 5: Ge 3i Ld Bf 8h Qc Vb

p = 6: Be Za Hd 6g 0h Nc Tb

p = 7: Ya Dd 6f Kc Rb

p = 8: 8e Xa 0f Hc Pb

p = 9: 3e Wa Ec Nb

p = 10: Va 8d Bc Lb

p = 11: Ua 4d Jb

p = 12: Ta 0d Hb

p = 13: Sa 9c Fb

p = 14: Ra 6c Db

p = 15: Qa 3c Bb

p = 16: Pa 0c

p = 17: Oa

p = 18: Na

p = 19: Ma

p = 20: La 8b

p = 21: Ka 6b

p = 22: Ja 4b

p = 23: Ia 2b

p = 24: Ha 0b

p = 25: Ga

p = 26: Fa

p = 27: Ea

p = 28: Da

p = 29: Ca

p = 30: Ba

p = 31: Aa