VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

KRY 1. Projekt 2023 / 24

Bc. Petr Pouč - xpoucp01

11. března 2023

Šifrování Afinitní šifry

Šifrování se nachází ve funkci `encode`. Vstupem funkce jsou klíče `a,b` a `text` k šifrování. Pro každý znak vstupní textu se nejprve zjistím jeho pozici v ASCII tabulce a následně pro každý znak aplikuji rovnici pro šifrování pomocí Afinitní šifry:

$$E(x) = (a * x + b) \mod 26$$

ASCII hodnoty jsou opět převedeny na znaky abecedy. Všechny mezery ve vstupním textu jsou během procesu šifrování ignorovány.

Dešifrování Afinitní šifry pomocí klíčů

Dešifrování se provádí ve funkci `decode`. Vstupní argumenty jsou stejné jako u předchozí funkce. Rovnice pro dešifrování vypadá následovně:

$$D(x) = a^{-1}(x–b) \mod 26$$

Multiplikativní inverze se počítá ve funkci `multiplicative_inverse`.

```
int multiplicative_inverse(int a, int m) {
    int MI = 0;
    for(unsigned i = 0; i < a; i++){
        MI = ((i * ALPHABET_SIZE) + 1);
        if (MI % a == 0){
            break;
        }
    }
    MI=MI/a;
    return MI;
}
```

Výstup této funkce představuje člen a^{-1}, který dále násobíme ASCII podobou znaku, od které je odečten klíč `b`, to celé dle vzorce modulo počtem znaků abecedy.

Dešifrování Afinitní šifry bez znalosti klíče

Implementace se nachází ve funkci `nokey_decode`. Dešifrování Afinitní šifry bez znalosti klíče je založeno na frekvenční analýze.

Klíč “a” může nabýt pouze hodnot 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23 a 25. Klíč “b” může nabýt hodnot 1 až 25. Celkem tedy existuje 12 * 25 kombinací klíčů. Rozhodl jsem se tedy zašifrovaný text dešifrovat pomocí funkce `decode` pomocí všech těchto kombinací klíčů.

Pro každý z těchto dešifrovaných textů spočítám frekvenční analýzu, která je implementována ve funkci `frequency`.

```
for(unsigned i = 0; i < ALPHABET_SIZE; i++){
    for(unsigned j = 0; j < strlen(s); j++){
        if(s[j] == 'A' + i || s[j] == 'a' + i) {count++;}
    }
    freq[i] = (float)count / strlen(s);
    count = 0;
}
return freq;
```

Funkce vrací výskyt jednotlivých znaků v šifrovaném textu jako procenta v desetinné podobě. Výskyt jednotlivých znaků české abecedy jsem si vytvořil v souboru `freq.h`.

```
static const float czech_freq[] = {
  0.115,   // a
  0.0187,  // b
  0.0187,  // c
  0.0125,  // d
  0.1975,  // e
  0.0094,  // f
  0.0242,  // g
  0.0242,  // h
  0.0911,  // i
  0.0019,  // j
  0.0405,  // k
  0.0361,  // l
  0.0345,  // m
  0.1045,  // n
  0.0582,  // o
  0.0405,  // p
  0.0002,  // q
  0.0563,  // r
  0.0497,  // s
  0.0626,  // t
  0.0304,  // u
  0.013,   // v
  0.0006,  // w
  0.0193,  // x
  0.0016,  // y
  0.0214,  // z
  };
```

Správný ze zašifrovaných textů poznám pomocí nejmenší hodnoty “odchylky”. To znamená, že hledám takový text, jehož frekvenční analýza se nejvíce podobá frekvenční analýze českých textů.

```
 for(int i = 0; i < A_KEYS; i++){
    for(int j = 0; j < B_KEYS; j++){
        char *decoded_text = decode(a_key[i], b_key[j], encoded_text);
        decoded_freq = frequency(decoded_text);
        error = 0;
        for(int k = 0; k < ALPHABET_SIZE; k++){
            error += fabs(czech_freq[k] - decoded_freq[k]);
        }

        if(error < min_error){
            min_error = error;
            right_a = a_key[i];
            right_b = b_key[j];
        }
    }
}
```

Listing 1: Nahrazování slov metodou synonym.

Pro každý text spočítám odchylku jako absolutní hodnotu rozdílu těhto frekvencí. Pro nejmenší odchylku frekvencí si zapamatuji hodnotu klíčů, pomocí kterých byl daný text dešifrován a tyto klíče vracím na vstup jako výsledek.