# Mendelova univerzita v Brně

# Provozně ekonomická fakulta

# 

## **Komprimace dat a kryptologie**

Implementace kompresních algoritmů

## Bc. Michal Melichar 18.01.2020

# Popis aplikace a uživatelská dokumentace

Aplikace implementuje kompresní algoritmy Huffmanovo kódování, Run-length encoding (RLE) a Lempel-Ziv-Welch. Uživatel zadá vstupní text, na nějž se po odeslání aplikují všechny tři algoritmy (každý ale samostatně). Pro každý algoritmus se provede nejdříve zakódování a následně se takhle zakódování řetězec aplikuje na dekódování. Ve výsledcích se zobrazí také porovnání velikosti výstupů z algoritmů a potřebná doba pro zakódování a dekódování. Pro každý algoritmus se navíc uživateli zobrazí užitečné informace.

## Zadání vstupních dat

Vzhled úvodní obrazovky lze vidět na obrázku číslo 1. Nachází se zde pole pro vložení vstupního textu a tlačítko „Odeslat“, které slouží k odeslání vstupního textu a provedení komprese zadaného vstupu. Aplikace je v čistém designu, který uživatele ničím neruší.

Obsah obrázku snímek obrazovky

Popis byl vytvořen automaticky

Obrázek - Úvodní obrazovka

## Zobrazení výsledků

Na vstup jsme zadali řetězec „Hello, worldddd“, ke kterému se vážou kapitoly 1.2.1, 1.2.2, 1.2.3 a 1.2.4.

### Graf výsledků

Na obrázku číslo 2 vidíme dva grafy, které nám porovnávají vlastnosti implementovaných kompresních algoritmů.

Graf na levé straně zobrazuje velikost vstupního textu, velikost textu po zakódování pomocí Huffmanovo kódování, Run Length Encoding a Lempel-Ziv-Welch kódování. Velikosti do grafu jsou vyneseny v bytech.

Graf vpravo naopak zobrazuje potřebné časy pro zakódování i dekódování pomocí zmíněných tří algoritmů. Potřebné časy jsou počítány v milisekundách.

Obsah obrázku snímek obrazovky

Popis byl vytvořen automaticky

Obrázek - Graf výsledků

### Huffmanovo kódování

Na obrázku číslo 3 můžeme vidět výstup pro algoritmus Huffmanova kódování. Výstup se skládá z částí:

* Výstupní tvar zprávy po jejím zakódování
* Velikost zakódované zprávy – každý znak je chápán jako jeden bit
* Výstupní tvar zprávy po jejím dekódování
* Velikost dekódované zprávy – každý znak je chápán jako jeden znak řetězce, potřebujeme tedy pro jeho uložení jeden byte
* Slovníku
* Informací, které porovnávají nezakódovanou zprávu a zakódovanou část
  + Progress bar znázorňuje poměr nezakódované zprávy a zakódované zprávy
  + Kompresní poměr k – vypočítán jako , uváděn jako desetinné číslo
  + Kompresní zisk z – vypočítán jako , uváděno jako desetinné číslo

Obsah obrázku snímek obrazovky

Popis byl vytvořen automaticky

Obrázek - Huffmanovo kódování

### Run Length Encoding

Na obrázku číslo 4 můžeme vidět výstup pro algoritmus Run Length Encoding. Výstup se skládá z částí:

* Výstupní tvar zprávy po jejím zakódování
* Velikost zakódované zprávy – každý znak je chápán jako jeden znak řetězce, potřebujeme tedy pro jeho uložení jeden byte
* Výstupní tvar zprávy po jejím dekódování
* Velikost dekódované zprávy – každý znak je chápán jako jeden znak řetězce, potřebujeme tedy pro jeho uložení jeden byte
* Informací, které porovnávají nezakódovanou zprávu a zakódovanou část
  + Progress bar znázorňuje poměr nezakódované zprávy a zakódované zprávy
  + Kompresní poměr k – vypočítán jako , uváděn jako desetinné číslo
  + Kompresní zisk z – vypočítán jako , uváděno jako desetinné číslo

Obsah obrázku snímek obrazovky

Popis byl vytvořen automaticky

Obrázek - Run Length Encoding

### Lempel-Ziv-Welch kódování

Na obrázku číslo 5 můžeme vidět výstup pro algoritmus Lempel-Ziv-Welch kódování. Výstup se skládá z částí:

* Výstupní tvar zprávy po jejím zakódování
* Velikost zakódované zprávy – Zde je několik možností, jak reprezentovat výslednou velikost. V této aplikaci byla zvolena možnost, čísla jsou chápána jako čísla, a nikoliv jako znaky. Pro uložení čísla byla použita konstanta PHP\_INT\_SIZE, následně byla vynásobena počtem číslic a vydělena 8 bity. Výstup je tedy brán jako pole číselných hodnot.
* Výstupní tvar zprávy po jejím dekódování
* Velikost dekódované zprávy – každý znak je chápán jako jeden znak řetězce, potřebujeme tedy pro jeho uložení jeden byte
* Slovníku
* Informací, které porovnávají nezakódovanou zprávu a zakódovanou část
  + Progress bar znázorňuje poměr nezakódované zprávy a zakódované zprávy
  + Kompresní poměr k – vypočítán jako , uváděn jako desetinné číslo
  + Kompresní zisk z – vypočítán jako , uváděno jako desetinné číslo

Obsah obrázku snímek obrazovky

Popis byl vytvořen automaticky

Obrázek - Lempel-Ziv-Welch kódování

## Omezení

Před samotným kódováním se provede optimalizace vstupu. Ta spočívá v nahrazení znaků s diakritikou na odpovídající znaky bez ní. Dále se odeberou čísla ze vstupu, jelikož ty by způsobovaly problémy pro algoritmus RLE. To z důvodu, že byl zvolený formát, který dokáže dosáhnout větší komprese, ale za následek menšího rozsahu používaných znaků na vstupu.

# Specifikace aplikace

## Požadavky na server

* PHP 7.3.8 a vyšší
* Připojení k internetu – z důvodu použití Google Charts

## Použité technologie

Aplikace je napsána v jazyce PHP verze 7.3.8 (nižší verze nebyly testovány) a využitím frameworku Nette 3. Pro grafickou podobu se využívá framework Bootstrap 4.4.1 a jQuery 3.4.1. Aplikace využívá AJAXu pro odesílání vstupního textu na server. Po odeslání vstupního dat a jejich zpracování, se dynamicky překreslí část obrazovky (pomocí snippetů), bez rušivého načtení celé stránky. Pro vykreslení dat do grafu je použita poslední verze Google Charts.

## Implementace

Aplikace využívá návrhového vzoru MVC, respektive MVP. Kdy vrstva V (view) se používá pro zobrazení dat uživateli a neprovádí žádnou manipulaci s daty. Vrstva M (model) se stará o výpočty aplikace, v našem případě tedy o kódování a nekódování pomocí patřičných algoritmů. Vrstva P(presenter) se stará o komunikaci mezi vrstvami view a model. Obsahuje také logiku, zda požadavek přišel pomocí AJAXu a podle toho připraví data pro vrstvu view.

Každý ze tří implementovaných kompresních algoritmů je zastoupen vlastní třídou. Pro Huffmanovo kódování se jedná o třídu Huffman, pro Run Length Encoding je to třída RunLength a pro LZW algoritmus se jedná o třídu LZW.

Formulář pro zadání dat odešle pomocí AJAXu na server data, ty zpracuje formulářová třída. Následně se provede běh kompresních algoritmů a server odesílá výsledky.

Při kódování i dekódování se počítá čas, která je potřebný pro danou operaci. Slouží k tomu funkce z PHP hrtime(), která vrací čas v nanosekundách. V čase běhu algoritmu není nejsou zahrnuty operace pro vytvoření analytických dat. Prostým součtem se tedy nedostane celkový čas běh programu.

### Třída Huffman

Třída obsahuje veřejné metody:

* constructor – Který jako parametr přijímá vstupní řetězec (tedy data, které mu odeslal formulář).
* encode – Funkce nemá žádné vstupní data. Nejdříve spočítá výskyty znaků v daném řetězci, poté se z nich vytvoří seřazené pole. Následně se vytváří zanořená posloupnost, která odpovídá postupnému vytváření sloupců, vždy o jeden menší, než je jeho předchozí a kde se sčítají poslední dvě nejmenší pravděpodobnosti. Následně se rekurzivně vygeneruje číselná posloupnost 0 a 1 pro každý použitý znak. Jako poslední krok se vezme vygenerovaná kódovací tabulka a pomocí ní se zakódují vstupní jednotlivé znaky vstupního řetězce.
* decode – Funkce přijímá jako vstupní parametr zakódovaný řetězec, tedy posloupnost nul a jedniček. Pomocí přechodové tabulky z něj pak sestaví odpovídající výstupní řetězec.
* getAnalysisData – Funkce slouží k vygenerování analytických dat, jako je velikost vstupu a výstupu, kompresní poměry, zisky a časy.

Dále obsahuje soukromé metody, které slouží jako pomocné – pro předzpracování dat a manipulaci s daty.

### Třída RunLength

Byl zde zvolen formát kódování takový, že pro 2 a více stejných znaků po sobě jdoucích, se na zapíše číslo, které odpovídá kolikrát se opakuje za sebou daný znak a následně se zapíše opakovaný znak. Absence uvozujícího znaku pro nahrazenou sekvenci má za výhodu větší kompresní poměr. Naopak nevýhodou je, že máme omezenější prostor použitelných znaků. Jelikož se poté nesmí ani číslice vyskytovat na vstupu pro kódování. Na výstupu by poté byly brány jako počet opakování následujícího znaku, nikoliv jako znak samotný.

Třída obsahuje veřejné metody:

* constructor – Který jako parametr přijímá vstupní řetězec (tedy data, které mu odeslal formulář).
* encode – Funkce nemá žádné vstupní data. Funkce využívá PHP metody preg\_replace\_callback, které má za cíl pomocí regulárního výrazu upravit vstup. Regulární výraz je zde „/(.)[\\1+/](file://1+/)“. Který pro každý druhý a další výskyt znaku jej nahrazuje počtem opakování.
* decode – Funkce přijímá jako vstupní parametr zakódovaný řetězec. Funkce je postavena na funkci PHP preg\_replace\_callback. Regulární výraz pro úpravu je „/(\d+)([^0-9])/“, který nahrazuje číslice znak tolikrát, jaká je jejich hodnota.
* getAnalysisData – Funkce slouží k vygenerování analytických dat, jako je velikost vstupu a výstupu, kompresní poměry, zisky a časy.

Dále obsahuje soukromé metody, které slouží jako pomocné – pro předzpracování dat a manipulaci s daty.

### Třída LZW

Třída obsahuje veřejné metody:

* constructor – Který jako parametr přijímá vstupní řetězec (tedy data, které mu odeslal formulář).
* encode – Funkce nemá žádné vstupní data. Nejdříve je vytvořena kódovací tabulka, která se naplní znaky od 0 do 255, jedná se o ASCII hodnotu daného znaku. Následně se prochází vstupní řetězec, hledá se, zda máme v přechodové tabulce odpovídající číselnou reprezentaci. Nejdříve se hledá pouze jeden znak, poté se rozšiřuje na posloupnost znaků a těm se přiřazuje odpovídající číselná reprezentace. Výsledkem je poté posloupnost čísel, která je v programu chápana jako pole čísel.
* decode – Funkce přijímá jako vstupní parametr zakódovaný řetězec. Provede se základní nastavení kódovací tabulky, poté se prochází vstupní řetězec a upravuje se kódovací tabulka podle posloupnosti, která se ještě v ní nenachází. Následně se vezme z kódovací tabulky pro hodnotu odpovídající reprezentace a tím se složí výsledná posloupnost znaků.
* getAnalysisData – Funkce slouží k vygenerování analytických dat, jako je velikost vstupu a výstupu, kompresní poměry, zisky a časy.

Dále obsahuje soukromé metody, které slouží jako pomocné – pro předzpracování dat a manipulaci s daty.