Kompresja danych - projekt 2

Bartosz Matyjasiak

25 stycznia 2021

Spis treści

1 Opis problemu **2**

2 Opis rozwiązania **3**
2.1 Pseudokod kodowania 3
2.2 Pseudokod dekodowania 3

3 Wyniki **4**
3.1 Analiza kodowania plików tekstowych 4
3.2 Analiza kodowania plików audio 5
3.3 Analiza kodowania plików graficznych 5

4 Podsumowanie **7**

1 Opis problemu

Zadanie polega na zrealizowaniu prostego kodera i dekodera wykorzystującego metodę LZ77 (dla ustalenia uwagi proszę stosować metodę w wariancie przedstawionym w materiałach wykładowych). Przyjmujemy następujące założenia:

1. Kodowane pliki mają strukturę bajtową (tzn. mogą zawierać dane z dowolnego problemu, w którym występuje co najwyżej **256** różnych „komunikatów").

2. Długość *bufora słownika* wynosi **256** bajtów, a *bufora wejściowego (look-ahead buffer)* **15** bajtów, tzn. pojedynczy *wskaźnik do słownika* ma rozmiar **2.5** bajta (**1** bajt reprezentujący położenie kopii fragmentu znalezionej w słowniku czyli *offset*, **0.5** bajta do zakodowania *długości kopii* oraz **1** bajt zawierający „komunikat" bezpośrednio po znalezionym fragmencie).

3. Rozwiązanie ma mieć postać DWÓCH gotowych do użycia funkcji/skryptów/aplikacji, tzn. *kodera* i *dekodera*.

4. Koder ma dwa argumenty wejściowe, tzn. (1) nazwa wraz z rozszerzeniem kodowanego pliku oraz (2) nazwa (wraz z dowolnie zdefiniowanym rozszerzeniem, na przykład **.xxx**) pliku zakodowanego, który ma być zapisany w tym samym folderze, co plik kodowany.

5. *Dekoder* ma również dwa argumenty wejściowe, tzn. (1) nazwa (wraz z zastosowanym rozszerzeniem, na przykład **.xxx**) dekodowanego pliku oraz (2) nazwa (wraz z zadanym rozszerzeniem) pliku rozkodowanego, który ma być zapisany w tym samym folderze, co plik dekodowany. Zrealizowane rozwiązania powinny być przetestowane na wybranych przykładach (np. plikach w formacie **.txt**).

2 Opis rozwiązania

Koder oraz dekoder zostały zaimplementowane w pliku `LZ77.py` w odpowiadających ich funkcjach `encode` oraz `decode`. By posłuzyć się skryptem należy wybrać jeden z argumentów `--encode` lub `--decode` oraz podać nazwę pliku wejściowego oraz wynikowego.

Przykładowe użycie skryptu do zakodowania pliku `test.txt`:

```
python LZ77.py --encode test.txt test.txt.lz77
```

Oraz zdekodowania go:

```
python LZ77.py --decode test.txt.lz77 test.txt.lz77.txt
```

Do projektu dołączyłem też skrypt bash `test.sh`, który zakoduje przykładowe pliki w folderze *input* i zapisze jest w folderze *lz77*, a następnie odkoduje je i zapisze w folderze *decoded*

2.1 Pseudokod kodowania

```
Dopóki w buforze wejściowym są wartości wykonuj:
  1. Wyszukaj w buforze słownikowym + buforze wejściowym najdłuższy
    ciąg wartości zaczynający się od początku bufora wejściowego,
    ciąg ten nie może mieć długość bufora wejściowego.
  2. Zapisujemy wynikową trójke (offset, characters_matching, lastchar)
    gdzie offset jest indeksem względem pozycji startu bufora wejściowego,
    characters_matching jest długością znalezionego ciągu wartości
    oraz lastchar jest następną wartością w buforze wejściowym po znalezionym ciąg
  3. Przesuń obydwa bufory o characters_matching+1.
```

2.2 Pseudokod dekodowania

```
Wypełnij bufor słownikowy wartościami początkowymi
Odkoduj z pliku wszystkie trójki (offset, characters_matching, lastchar)
Dla każdej z nich wykonuj:
  Dopóki characters_matching > 0 wykonuj:
    Skopiuj z bufora wartość z pozycji offset od końca na koniec bufora
    Zapisz ostatnią wartość z bufora jako część wyniku
    Jeżeli bufor jest większy od maksymalnego rozmiaru bufora:
```

```
    Przesuń wartości w buforze o jedną pozycje w lewo
    Usuń wartość końcową z bufora
  Zmniejsz wartość characters_matching o jeden

Skopiuj wartość lastchar na koniec bufora
Zapisz ostatnią wartość z bufora jako część wyniku
Jeżeli bufor jest większy od maksymalnego rozmiaru bufora:
  Przesuń wartości w buforze o jedną pozycje w lewo
  Zsuń wartość końcową z bufora
```

3 Wyniki

3.1 Analiza kodowania plików tekstowych

Do przetestowania kodera użyłem dwóch tekstów:

- Lokomotywa - Julian Tuwin
- Inwokacja - Pan Tadeusz - Adam Mickiewicz

Po uruchomieniu skryptu otrzymałem wyniki widoczne w tabelach 1 oraz 2.

Lokomotywa - Julian Tuwin	
stopień kompresji	1,309
procent kompresji	23,60%
średnia bitowa	6,112

Tablica 1: Wyniki zakodowania tekstu Lokomotywy Juliana Tuwima

Inwokacja - Pan Tadeusz - Adam Mickiewicz	
stopień kompresji	1,160
procent kompresji	13,82%
średnia bitowa	6,894

Tablica 2: Wyniki zakodowania tekstu Inwokacji Pana Tadeusza Adama Mickiewicza

Kodowanie lz77 sprawdziło się w kompresji plików tekstowych. W obydwu plikach średnia bitowa jest mniejsza od 8, a całe pliki zmniejszyły się o ok. 24% oraz ok. 14%. Skompresowany tekst wiersza Juliana Tuwima - Lokomotywa wypadł dużo lepiej. Bardzo możliwe z przyczyny iż wiersz ten zawiera bardzo dużo rymów i powtarzających się wyrazów. Dzięki temu koder może zaoszczędzić jeszcze więcej miejsca.

3.2 Analiza kodowania plików audio

Do przetestowania kodera użyłem dwóch plików audio:

- icing.wav
- song.wav

Po uruchomieniu skryptu otrzymałem wyniki widoczne w tabelach 3 oraz 4.

icing.wav	
stopień kompresji	0,761
procent kompresji	-31,32%
średnia bitowa	10,506

Tablica 3: Wyniki zakodowania pliku audio icing.wav

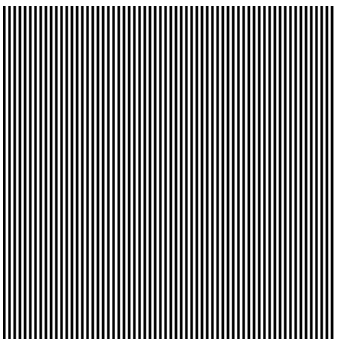
song.wav	
stopień kompresji	0,762
procent kompresji	-31,31%
średnia bitowa	10,505

Tablica 4: Wyniki zakodowania pliku audio song.wav

W przypadku plików audio koder nie jest najlepszym wyborem. Obydwa pliki zwiększyły swój rozmiar o około 30%.

3.3 Analiza kodowania plików graficznych

Do przetestowania kodera użyłem trzech obrazów: Po uruchomieniu skryptu otrzymałem wyniki widoczne w tabelach 5, 6 oraz 7.



Rysunek 1: img1.bmp

Rysunek 2: lena.png

Rysunek 3: moon.jpg

img1.bmp	
stopień kompresji	5,442
procent kompresji	81,63%
średnia bitowa	1,470

Tablica 5: Wyniki zakodowania pliku img1.bmp

lena.png	
stopień kompresji	0,657
procent kompresji	-52,18%
średnia bitowa	12,174

Tablica 6: Wyniki zakodowania pliku lena.png

moon.jpg	
stopień kompresji	0,669
procent kompresji	-49,52%
średnia bitowa	11,961

Tablica 7: Wyniki zakodowania pliku moon.jpg

Dla obrazów *lena.png* oraz *moon.jpg* wyniki kompresji są niezadowalające. Pliki tych obrazów zwiekszyły się o około 50%. Ciekawym przypadkiem jest obraz *img.bmp*, który zmniejszył się aż o 81%. Jednak takie obrazy rzadko są spotykane w codziennym użytku.

4 Podsumowanie

Koder LZ77 najlepiej nadaje się do kodowania tesktu oraz obrazów z dużą powtarzalnością kolorów pikseli. Nie nadaje się natomiast do kodowania plików audio czy też obrazów o dużej ilości szczegółów i kolorów jak portrety i krajobrazy.