Marcin Januszkiewicz
Raport z zadania implementacyjnego.

1 Wstęp

Celem zadania było napisanie implementacji algorytmów LZ77 i LZSS z dodaną kompresją Huffmana. Pondato należało zapewnić szybkie przeszukiwanie słownika.

2 Implementacja warstw LZ77 i LZSS

Algorytm LZSS polega na trzymaniu bufora ostatnio zakodowanych symboli i wyszukiwaniu w tym buforze najdłuższego podciągu będącego prefiksem bufora wejściowego. Na wyjście wypisywana jest trójka składająca się z pozycji i długości znalezionego podciągu oraz pierwszej litery której nie udało się zakodować. Do LZSS dodana jest optymalizacja która polega na wypisywaniu albo pary charakteryzującej ciąg, albo pojedynczego znaku. Pozwala to na pominięcie krótkich ciągów których kod byłby dłuższy niż wypisanie liter osobno.

W celu przyspieszenia przeszukiwania bufora utrzymywana jest struktura tablicy haszującej indeksowanej trójkami symboli, której wartości są zbiorami indeksów w buforze słownikowym. Pozwala to na szybkie wyznaczenie możliwych początków długich prefiksów.

3 Implementacja warstwy Huffmana

Krotki symboli powstały w wyniku kodowań LZ kompresujemy dodatkowo statycznym Huffmanem. Dla każdego możliwego elementu krotki (w LZ77 - pozycji w buforze, długości i znaku, w LZSS - znacznika, pozycji, długości i znaku) utrzymujemy osobne drzewo Huffmana. Pozwala to na osobną kompresję części danych o różnych rozkładach i wiedzy o występowaniu symboli w pliku do lepszej kompresji niż utworznie jednego drzewa dla wszystkich symboli, co oznaczałoby, że określone symbole moglyby wystąpić w dowolnym miejscu w pliku. Wykorzystanie statycznej wersji kodowania Huffmana powoduje, że musimy trzymać cały plik po kompresji LZ w pamięci przed wypisaniem.

4 Uwagi do implementacji

Algorytmy sa parametryzowane rozmiarami bufora słownikowego i wejściowego, ale te dane nie są zawarte w skompresowanym pliku, więc dekompresując plik trzeba znać te parametry.

Przy pierwszej próbie wykorzystania tablicy haszującej to pamiętania pozycji trójek pojawił się problem dużego narzutu na utrzymanie struktury, w szegółności na usuwanie nieaktualnych wpisów. W bieżącej implementacji wpisy usuwane są leniwie przy odwiedzaniu elementów tablicy, co może prowadzić do zajmowania dużej ilości pamięci i co za tym idzie przestojów w działaniu programu podczas uruchamiania garbage collectora. W przeciwieństwie do pierwszej implementacji która spowalniała działanie programu dwukrotnie, ta struktura przyspiesza działanie programu kilkukrotnie, co jest szczególnie widoczne przy większych słownikach.

5 Wyniki

W folderze `wyniki/details` znajduję się seria plików o nazwach a-b.txt. Są to szegółowe wyniki testów dla korpusów Cantenbury i Calgary z parametrami a - bufor słownika oraz b - bufor wejścia. W folderze `wyniki` znajdują się obrazki porównujące wpływ parametrów na statystyki kompresji tekstu `alice29.txt`, będącego angielskim tekstem książki Ąlicja w krainie czarów". Ponadto w pliku `special` znajdują się wyniki dla pewnych konkretnych plików. Oczywista uwaga jest taka, że charakterystyka danych ma duży wpływ na stopień kompresji i na wpływ zmiany parametrów na stopień kompresji. Ogólny trend jaki można zauważyć jest taki, że od pewnego momentu zwiększanie buforów przestaje być opłacalne. Być może pozwala nam to na znajdywanie dłuższych dopasowań ale w zakodowanym strumieniu przed huffmanem zwiększa nam rozmiar każdego symbolu, a ponadto może zmienić rozkład prawdopodobieństwa danych dla huffmana co zwiększy nam średnią długość kodu. Patrząc na statystyki można też szacować jakie najczęściej pojawiają sie długości dopasowań i pozycje w słowniku.