

Kompresja tekstu Metodą Huffmana

Opis tematyki projektu

Algorytm Huffmana jest algorytmem kompresji tekstu. Jego głównym założeniem jest, by litery występujące częściej miały krótszy kod, a te występujące rzadziej – dłuższy. Kod powstaje na podstawie drzewa, które w liściach posiada znaki występujące w tekście, który chcemy skompresować. Drzewo to jest skonstruowane w taki sposób, by długość drogi do liścia, w którym zapisany jest dany znak była uzależniona od ilości wystąpień danego znaku w histogramie. Pozwala to uzyskać kody krótsze niż 1 bajt dla często występujących w tekście znaków, co w efekcie przekłada się na kompresję tekstu.

Instrukcja kompilacji

Program został przetestowany na kompilatorach gcc oraz clang z następującymi flagami: -Wall -Wextra -pedantic -std=c99 -lm -O2

Instrukcja uruchomienia

Program uruchamiamy z poziomu linii komend, z jednym z następujących argumentów:

-e [plik wejściowy] [plik wyjściowy] – koduje plik wejściowy, i zapisuje go do pliku wyjściowego

-s [string] [plik wyjściowy] -koduje string i zapisuje go do pliku wyjściowego

-d [plik wejściowy] [plik wyjściowy] [plik z kluczem] – dekoduje plik wejściowy przy pomocy klucza wygenerowanego argumentami '-e' bądź '-s' i zapisuje wynik do pliku wyjściowego

-a [plik wejściowy] [plik wyjściowy] – wykonuje cały algorytm – tj. kodowanie pliku wejściowego, dekodowanie i zapis wyniku do pliku wyjściowego.

Opis fragmentów implementacji

generateTree() - Funkcja przyjmuje wskaźnik na korzeń drzewa i tablicę histogramu posortowaną malejąco po ilościach wystąpień. W pierwszym kroku znajduje ostatni element histogramu mający ilość wystąpień większą od zera. Następnie pobiera ten element oraz element przedostatni, tworzy dla nich węzły i rodzica i szuka drzewa, do którego mogłaby podłączyć powstałe drzewo. Jeśli znajdzie wskaźnik na drzewo, funkcja złączy te drzewa tworząc nowego rodzica, którego lewy wskaźnik wskazuje na nowo powstałe drzewo, a prawy na drzewo, do którego chcemy się podłączyć. Adres nowego rodzica ustawia jako korzeń i pobiera dwa kolejne elementy z histogramu, aż do wykorzystania wszystkich elementów. Jeśli jednak funkcja nie znajdzie drzewa, do którego mogłaby połączyć nowo powstałe drzewo to ustawia korzeń na nowo powstałe drzewo i pobiera kolejne elementy z histogramu. Funkcja zwraca wskaźnik na korzeń wygenerowanego w ten sposób drzewa.

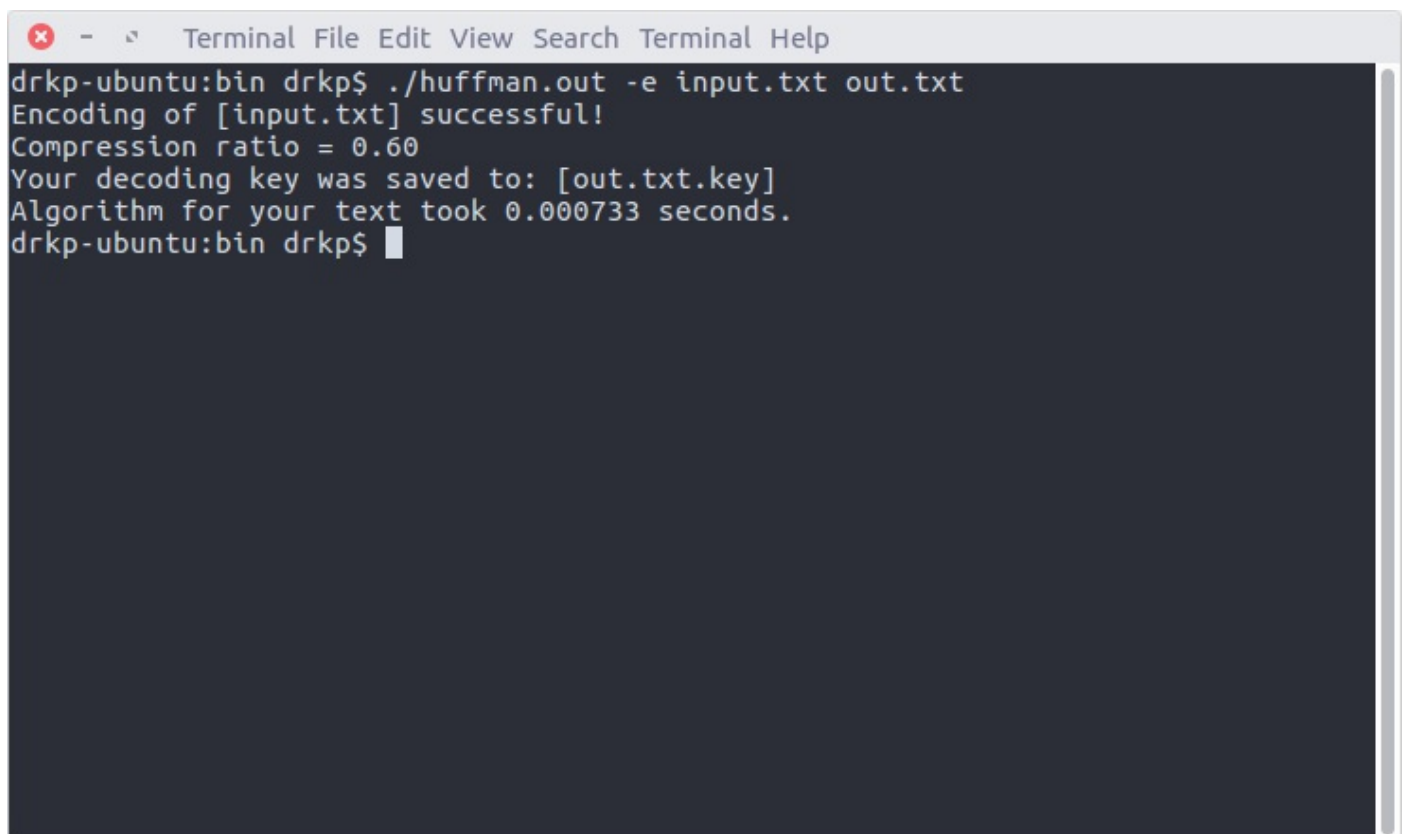
encode() - Funkcja pobiera z pliku wejściowego znak, znajduje jego kod w tablicy `int **codes` i zapisuje do buffera (tablica dziewięcioelementowa). Jeśli kod nie wypełni wszystkich pól buffera, pobierany jest następny znak. Jeśli buffer jest pełny jest on zamieniany na kod ASCII i zapisywany

do pliku wyjściowego. Pętla trwa aż do pobrania wszystkich znaków z pliku wejściowego. Funkcja zwraca poziom kompresji jako stosunek rozmiaru wyjściowego do wejściowego w formacie double.

decode() - Celem funkcji jest zamiana kodu zakodowanego w postaci kodów ASCII na kod binarny. Korzysta przy tym z pomocy funkcji `asciiToBin()`

Ponadto został wykorzystany algorytm quicksort w celu posortowania histogramu.

Zrzuty ekranu

A screenshot of a terminal window with a dark background and light-colored text. The window has a title bar with a red close button, a maximize button, and a menu bar containing 'Terminal', 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal content shows a user running a command to execute a Huffman encoding program. The output indicates successful encoding, a compression ratio of 0.60, the saving of a decoding key, and the execution time of 0.000733 seconds. The prompt returns to the user.

```
drkp-ubuntu:bin drkp$ ./huffman.out -e input.txt out.txt
Encoding of [input.txt] successful!
Compression ratio = 0.60
Your decoding key was saved to: [out.txt.key]
Algorithm for your text took 0.000733 seconds.
drkp-ubuntu:bin drkp$
```

```
drkp-ubuntu:bin drkp$ ./huffman.out -d out.txt dec.txt out.txt.key
Decoding of file [out.txt] successful!
Algorithm for decoding your text took 0.000389 seconds.
drkp-ubuntu:bin drkp$
```

Opis sposobu mierzenia wydajności programu

Wydajność programy mierzymy dwoma wartościami - miarą kompresji oraz pomiarem czasu. Kompresje mierzymy dzieląc rozmiar pliku skompresowanego przez plik wejściowy co zwraca nam dokładny poziom kompresji danych. Czas mierzymy za pomocą biblioteki `time.h` i funkcji `clock()`, która zwraca aktualny tick procesora. Dzięki dzieleniu przez stałą `"CLOCKS_PER_SEC"` otrzymujemy czas w milisekundach.

Wyniki przeprowadzonych testów



Wnioski

Algorytm Huffmana to bezstratny algorytm kompresji danych, który najlepiej sprawdza się przy tekście. W tekście dzięki niemu potrafimy uzyskać średnio do 26% kompresji, co więcej działa on szybko, kompresując plik o rozmiarze ~1MB w czasie 0,673s. W projekcie udało się, oprócz samej kompresji zrealizować również dekompresję oraz interfejs CLI. Najważniejszym elementem dalszego rozwoju projektu jest rozszerzenie działania programu poza zakres kodów ASCII oraz więcej usprawnień interfejsu użytkownika.

Autorzy

Maciej Brzęczkowski Mariusz Lewczuk