POLITECHNIKA POZNAŃSKA

WYDZIAŁ ELEKTRYCZNY

Instytut Automatyki, Robotyki i Inżynierii Informatycznej

Teoria informacji i kodowanie

Autorzy :

Cezary Czekalski 126907

Marcin Hilt 116854

Jan Śmigielski 111053

Styczeń 2020

Spis treści

1 Cel projektu **3**

2 Użyte narzędzia **3**

3 Opis użytego algorytmu **3**

3.1 Przykład użycia algorytmu **4**

3.2 Zalety oraz wady **8**

4 Porównanie do rozwiązań komercyjnych **8**

4.1 Porównanie rozmiaru pliku po kompresji **8**

5 Wnioski **10**

6 Źródła **10**

1 Cel projektu

Celem projektu jest przedstawienie i zaimplementowanie kompresji oraz dekompresji przy wykorzystaniu algorytmu statycznego kodowania Huffmana, obliczenie statystyk takich jak : prawdopodobieństwo, ilość wystąpień, jednostkę informacji czy entropię oraz porównanie wyników zaimplementowanej przez nas metody kompresji do wyników komercyjnych programów dostępnych na rynku.

2 Użyte narzędzia

W celu wykonania projektu wykorzystano język programowania Java 8 oraz środowisko programistyczne Intellij IDEA 2018.3.6 . Do wykonania interfejsu dla użytkownika wykorzystano Intellij IDEA Scene Builder extension.

3 Opis użytego algorytmu

Kodowanie Huffmana (ang. *Huffman coding*) – jedna z najprostszych i łatwych w implementacji metod kompresji bezstratnej. Została opracowana w 1952 roku przez Amerykanina Davida Huffmana.

Algorytm Huffmana nie należy do najefektywniejszych obliczeniowo systemów bezstratnej kompresji danych, dlatego też praktycznie nie używa się go samodzielnie. Często wykorzystuje się go jako ostatni etap w różnych systemach kompresji, zarówno bezstratnej, jak i stratnej, np. MP3 lub JPEG. Pomimo że nie jest doskonały, stosuje się go ze względu na prostotę oraz brak ograniczeń patentowych. Jest to przykład wykorzystania algorytmu zachłannego.

Kodowanie Huffmana polega na utworzeniu słów kodowych (ciągów bitowych), których długość jest odwrotnie proporcjonalna do prawdopodobieństwa p_i . Tzn. im częściej dany symbol występuje (może wystąpić) w ciągu danych, tym mniej zajmie bitów.

Własności kodu Huffmana są następujące:

- jest kodem prefiksowym; oznacza to, że żadne słowo kodowe nie jest początkiem innego słowa;
- średnia długość słowa kodowego jest najmniejsza spośród kodów prefiksowych;
- jeśli prawdopodobieństwa są różne, tzn. $p_j > p_i$, to długość kodu dla symbolu x_j jest nie większa od kodu dla symbolu x_i słowa kodu dwóch najmniej prawdopodobnych symboli mają równą długość.

Kompresja polega na zastąpieniu symboli otrzymanymi kodami.

W naszym projekcie została użyta statyczna odmiana tego kodowania sposób jego działania został przedstawiony poniżej :

1. Określamy prawdopodobieństwo(lub częstość występowania) dla każdego symbolu z pliku wejściowego .
2. Tworzymy listę drzew binarnych, które w węzłach przechowują pary: symbol, prawdopodobieństwo. Na początku drzewa składają się wyłącznie z korzenia.

3. Dopóki na liście jest więcej niż jedno drzewo, powtarzamy następujące kroki :
    1. Usuwamy z listy dwa drzewa o najmniejszym prawdopodobieństwie zapisanym w korzeniu.
    2. Wstawiamy nowe drzewo, w którego korzeniu jest suma prawdopodobieństw usuniętych drzew, natomiast one same stają się jego lewym i prawym poddrzewem. Korzeń drzewa nie przechowuje symbolu.

Drzewo, które pozostanie na liście, jest nazywane drzewem Huffmana – prawdopodobieństwo zapisane w korzeniu jest równe 1, natomiast w liściach drzewa zapisane są symbole.

Na podstawie drzewa Huffmana tworzone są słowa kodowe; algorytm jest następujący:

1. Każdej lewej krawędzi drzewa przypisujemy 0, prawej 1 (można oczywiście odwrotnie).
2. Przechodzimy w głąb drzewa od korzenia do każdego liścia (symbolu):
    1. Jeśli skręcamy w prawo, to dopisujemy do kodu bit o wartości 1.
    2. Jeśli skręcamy w lewo, to dopisujemy do kodu bit o wartości 0.

Długość słowa kodowego jest równa głębokości symbolu w drzewie a wartość binarna zależy od jego położenia w drzewie

Dekompresja wymaga posiadania drzewa kodów, a odtworzenie danych polega na czytaniu kolejnych bitów ze strumienia danych i jednoczesnym przechodzeniu drzewa zgodnie z przyjętą konwencją (lewo/prawo). Po dojściu do liścia i wypisaniu związanego z nim symbolu powracamy do korzenia.

Statyczny algorytm kodowania Huffmana jest algorytmem niedeterministycznym, ponieważ nie określa, w jakiej kolejności wybierać drzewa z listy, jeśli mają takie samo prawdopodobieństwo. Nie jest również określone, które z usuwanych drzew ma stać się lewym bądź prawym poddrzewem. Jednak bez względu na przyjęte rozwiązanie średnia długość kodu pozostaje taka sama.

3.1 Przykład użycia algorytmu

Do przedstawienia kodowania przy użyciu algorytmu statycznego kodowania Huffmana wykorzystano plik wejściowy o zbiorze symboli i ich częstości występowania zgodnymi z tymi zawartymi w tabeli 1.

Tabela 1: Symbole i ich częstość występowania

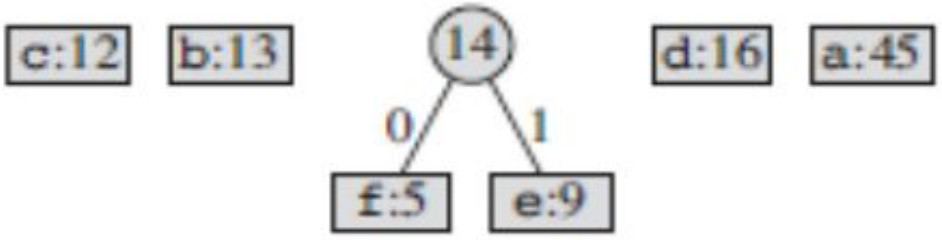
Symbol	Częstość występowania
a	45
b	13
c	12

d	16
e	9
f	5

Na początku należy zbudować listę drzew składającą się wyłącznie z korzeni:

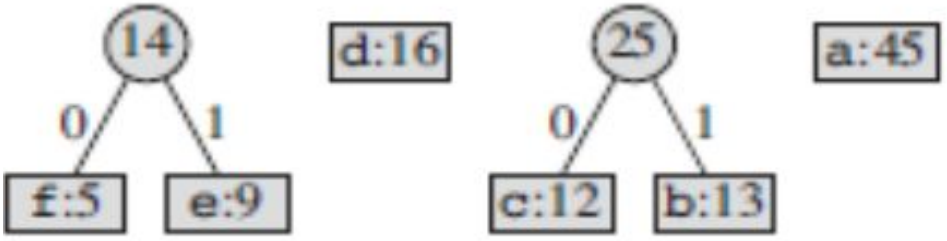
Rysunek 1: Lista drzewa z korzeni

Kolejnym krokiem jest wybranie z tej listy dwóch drzew o najmniejszej liczbie wystąpień (w korzeniu) i zastąpienie ich drzewem dla którego stają się odpowiednio lewym i prawym synem. Korzeń nowego drzewa przyjmuje ilość wystąpień równą sumie liczby wystąpień w korzeniu "usuwanych" drzew. W naszym wypadku są drzewa z symbolami “f” o ilości wystąpień 5 oraz “e” o ilości wystąpień 9. Drzewo o mniejszej liczbie wystąpień staje się liściem lewym, a drzewo większej liściem prawym.



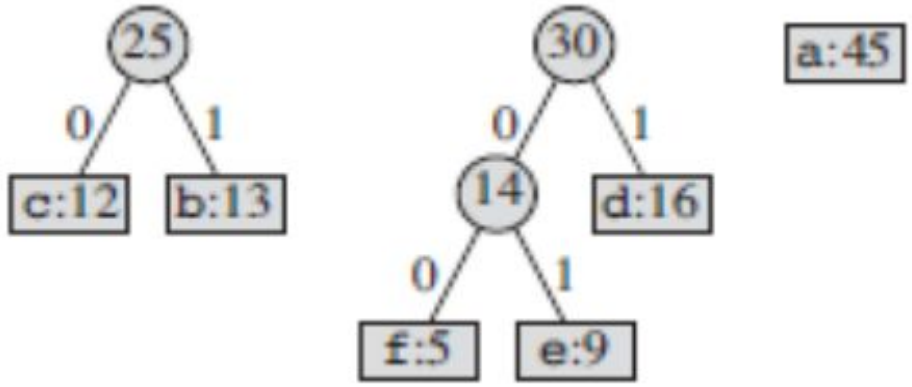
Rysunek 2 : Drzewo po połączeniu dwóch drzew o najmniejszej liczbie wystąpień

Kontynuujemy dalsze wybieranie drzew o najmniejszej liczbie wystąpień i zastąpienie ich drzewem. Nowe drzewa o najmniejszej liczbie wystąpień w korzeniu to c (12) oraz b (13) a więc nowo powstałe drzewo z sumą ich wystąpień to 25.

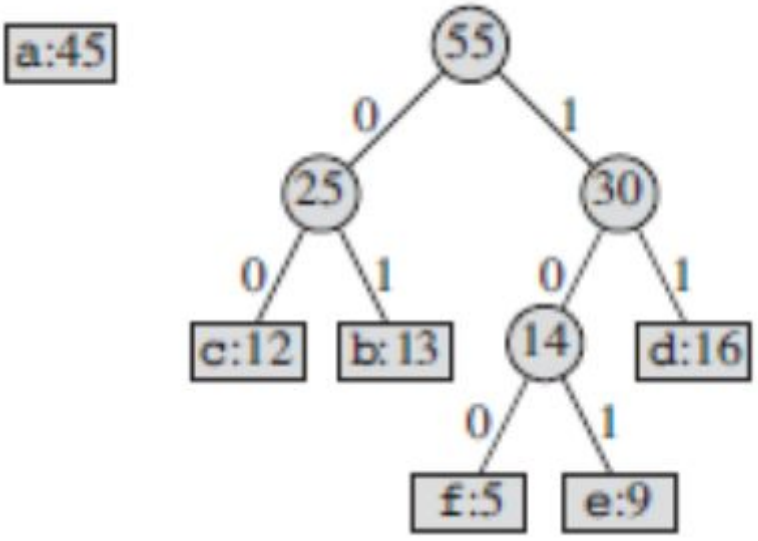


Rysunek 3 : Kolejny etap kodowania

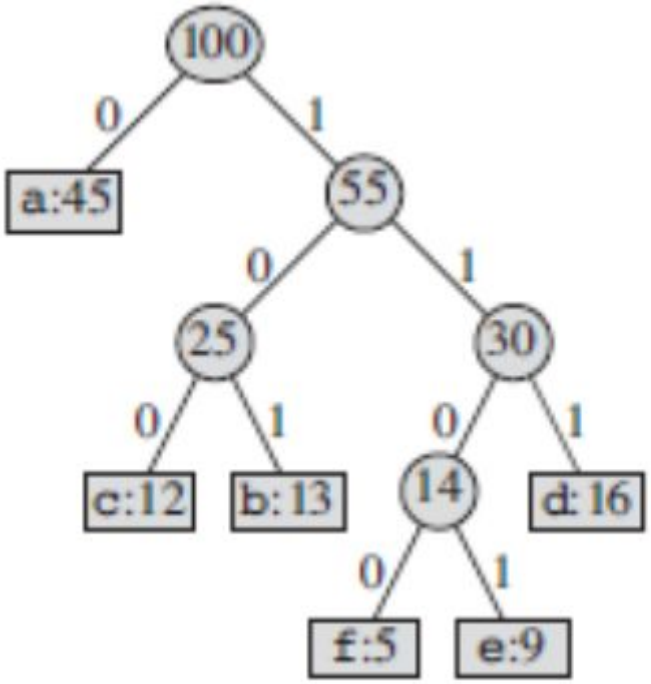
Kontynuujemy zgodnie z powyżej opisanym schematem do momentu gdy na liście pozostanie dokładnie jeden element -- korzeń drzewa kodów. Liczba wystąpień zapisana w tymże korzeniu jest oczywiście równa sumie liczbie wystąpień wszystkich korzeni, czyli w naszym wypadu 100. Na rysunkach 4 i 5 przedstawiono kolejne etapy a na rysunku 6 przedstawiono ostatecznie otrzymane drzewo.



Rysunek 4 : Kolejny etap kodowania połączenie korzenie 14 i 16



Rysunek 5 : Kolejny etap kodowania połączenie 30 i 25



Rysunek 6 : Połączenie korzeni 45 i 55 otrzymanie korzenia drzewa kodów

W tabeli 2 przedstawiono kod jaki będzie przypadał każdemu symbolowi oraz długość tego kodu w bitach. Kod wyznacza się przechodzą po drzewie więc jeżeli chcemy uzyskać kod dla znaku c musimy : przejść z korzenia 100 do liścia 55 przechodzimy w prawo a więc zgodnie z ustalonym porządkiem do kodu wpisujemy 1. Następnie przechodzimy z liścia 55 do liścia 25 - wpisujemy 0 i następnie do liścia w którym znajduje się c czyli znowu 0 a więc otrzymujemy kod 100.

Tabela 2 : Dodano

Symbol	**Częstość występowania**	**Kod**	**Długość kodu [bity]**
a	45	0	1
b	13	101	3
c	12	100	3
d	16	111	3
e	9	1101	4
f	5	1100	4

Jak widać element który występuje najczęściej ma jednocześnie najkrótszy kod.

Dekompresja wymaga posiadania drzewa kodów, a odtworzenie danych polega na czytaniu kolejnych bitów ze strumienia danych i jednoczesnym przechodzeniu drzewa zgodnie z przyjętą konwencją (lewo/prawo). Po dojściu do liścia i wypisaniu związanego z nim symbolu powracamy do korzenia. Np. ciąg bitów 10011111000 koduje cdfa.

3.2 Zalety oraz wady

W tym rozdziale w tabeli nr 3 przedstawione zostały zalety oraz wady statycznego kodowania Huffmana.

Tabela 3: Zalety oraz wady statycznego kodowania Huffmana

Zalety	Wady
jest optymalnym kodem prefiksowym	do budowy drzewa konieczne są statystki kodowanej wiadomości
procedura budowania drzewa jest szybka i prosta w implementacji	do przekazywanej / zapisywanej wiadomości trzeba dołączyć opis drzewa
zarówno kodowanie jak i dekodowanie jest proste i efektywne	

4 Porównanie do rozwiązań komercyjnych

W tym rozdziale przedstawione zostanie porównanie naszej implementacji algorytmu kodowania Huffmana metoda statyczną do programów komercyjnych dostępnych na rynku : Winrar w wersji 5.70 wykorzystującego do kompresji rar oraz zip i 7zip w wersji 19.0 wykorzystującego do kompresji zip oraz 7z

4.1 Porównanie rozmiaru plików po kompresji

Porównania działania kompresji z programami komercyjnymi wykonano na 3 plikach tekstowych których zawartość została wygenerowana z tekstu Pana Tadeusza.

W następnych 3 tabelach znajduje się porównanie dla różnych programów dla kolejno 1000 słów w tabeli nr 4 ,5000 w tabeli nr 5 i 10000 słów w tabeli nr 6. SK oznacza stopień kompresji. W przypadku programów Winrar i 7zip SK jest liczone poprzez podzielenie liczby bitów po kompresji do liczby bitów przed kompresją a następnie mnożone przez 100 i wyrażane w %.

$$SK = \frac{\textit{wielkość pliku po kompresji}}{\textit{wielkość pliku przed kompresją}} * 100$$

Tabela 4 : Porównanie stopnia kompresji dla 1000 słów

Program	Ilość słów	wielkość pliku przed kompresją [bajty]	wielkość pliku po kompresji [bajty]	SK [%]
Nasza implementacja	1000	6768	4250	62,8
Winrar 5.7 zip	1000	6768	3428	50,7
Winrar 5.7 rar	1000	6768	3408	50,4
7zip 19.0 zip	1000	6768	3396	50,2
7zip 19.0 7zip	1000	6768	3425	50,6

Tabela 5 : Porównanie stopnia kompresji dla 5000 słów

Program	Ilość słów	wielkość pliku przed kompresją [bajty]	wielkość pliku po kompresji [bajty]	SK [%]
Nasza implementacja	5000	34 285	18 857	55,0
Winrar 5.7 zip	5000	34 285	3 774	11,0
Winrar 5.7 rar	5000	34 285	3 483	10,1
7zip 19.0 zip	5000	34 285	3 748	10,9
7zip 19.0 7zip	5000	34 285	3 531	10,3

Tabela 6 : Porównanie stopnia kompresji dla 10000 słów

Program	Ilość słów	wielkość pliku przed kompresją [bajty]	wielkość pliku po kompresji [bajty]	SK [%]
Nasza implementacja	10000	68 240	37 464	54,9
Winrar 5.7 zip	10000	68 240	4072	6,0
Winrar 5.7 rar	10000	68 240	3501	5,1
7zip 19.0 zip	10000	68 240	4039	5,9

7zip 19.0 7zip	10000	68 240	3558	5,2

5 Wnioski

Kodowanie Huffmana metodą statyczną jest jedną z najprostszych i łatwych w implementacji metod kompresji bezstratnej, nie należy on jednak do najefektywniejszych obliczeniowo systemów bezstratnej kompresji danych, co zostało wykazane przy porównaniu naszej implementacji tej metody do komercyjnych programów dostępnych na rynku. Przy najmniejszym pliku tekstowym różnica w stopniu kompresji była stosunkowo niewielka około 12% a różnica w wielkości pliku od najlepiej skompresowanego pliku wynosiła 854 bajtów. Już w wypadku 5000 słów widać bardzo duża różnica w kompresji, aż 49%. Różnica między programem o najlepszej kompresji a naszą implementacją wynosiła, aż 15 374 bajtów. A w przypadku 10000 słów jest to prawie 50% a różnica wynosi, aż 33 963 bajtów. W obliczu powyższych wyników można zauważyć bardzo dużą różnice między naszą implementacją metody statycznej Huffmana a programami komercyjnymi korzystającymi z metod kompresji : zip, 7z oraz rar. Największy wpływ ma na to wybór metody która tak jak wcześniej było wspomniane nie należy do najefektywniejszych obliczeniowo systemów bezstratnej kompresji danych.

6 Źródła

W tym rozdziale zostaną przedstawione źródła z których korzystaliśmy tworząc projekt.

[1] Cormen-Introduction To Algorithms

[2] D.Duda - Kodowanie Huffmana