REPREZENTACJA INFORMACJI

1 Cel ćwiczenia

Ćwiczenie ma na celu omówienie wybranych koderów bezstratnych, zastosowanie ich do kodowania różnych typów danych oraz eksperymentalne zbadanie wpływu ich zastosowania na redukcję nadmiarowości.

2 Dobór danych testowych

W ćwiczeniu zostały użyte różnego typu dane: tekst, dźwięk, wideo i obraz. Po pobraniu niektóre źródła zostały wypakowane aby uzyskać docelowy format plików. Poniżej znajduje się tabela zawierająca dane źródłowe.

Table 1: Lista użytych źródeł

Name	Type	Properties	Size [MB]	Link
crew	film	YUV4MPEG, 60 fps, 704x576	349.0	https://media.xiph.org/video/derf/y4m/crew_4cif.y4m
waterfall	film	YUV4MPEG , 30 fps, 352x288	38.0	https://media.xiph.org/video/derf/y4m/waterfall_cif.y4m
station2	film	YUV4MPEG, 25 fps, 1920x1080	930.0	https://media.xiph.org/video/derf/y4m/station2_1080p25.y4m
photo set	zdjęcie	RGB 16 Bit	897.0	http://imagecompression.info/test_images/rgb16bit.zip
bach	dźwięk	PCM, stereo, 44.1kHz	2.0	http://soundexpert.org/documents/10179/13123/bah.rar/385ddfaa-4738-4294-a23e-597c47b934ac
quartet	dźwięk	PCM, stereo, 44.1kHz	2.0	http://soundexpert.org/documents/10179/13123/qrt.rar/6a1b25c1-24aa-43e7-a58f-3a8e0d3f160b
speech	dźwięk	PCM, stereo, 44.1kHz	2.0	http://soundexpert.org/documents/10179/13123/fms.rar/4d6c8665-a009-474f-9121-cb676fd19f56
enwik8	tekst	ascii	100.0	http://mattmahoney.net/dc/enwik8.zip
fork	tekst	sourcecode	0.7	https://raw.githubusercontent.com/torvalds/linux/master/kernel/fork.c

3 Zbadane kodery

Do przeprowadzenia eksperymentu wybrano popularne kodeki, które potrafią kodować co najmniej jeden z badanych typów danych.

kodek	tekst	obraz	wideo	dźwięk
Deflate	+	+	+	+
Bzip2	+	+	+	+
LZMA	+	+	+	+
FLAC				+
ALAC				+
PNG		+		
WEBP		+		
H.264			+	
H.265			+	
FFV1			+	

4 Kodowanie tekstu

Sprawdzenie wpływu kodera na dane polegało na zastosowaniu jego do zakodowania każdego zestawu danych. Ponieważ wszystkie informacje używane przez dzisiejsze komputery są zapisane w postaci binarnej można je poddać kompresji algorytmami ogólnego użytku, które nie biorą pod uwagę semantyki kompresowanych danych.

Do kompresji został użyty program 7-zip. Dla poszczególnych kodowań zostały użyte poniższe ustawienia.

kodek	format	rozmiar słownika	rozmiar słowa
Deflate	gzip	32 KB	128
BZip2	bzip2	900 KB	-
LZMA	zip	64 MB	64

5 Wyniki kodowania tekstu

Poniżej zostały przedstawione rezultaty użycia poszczególnych koderów.

Table 4: Kompresja Deflate - gzip

Name	Format before	Size before	Size after	ratio [%]
crew	YUV4MPEG	364957238	205170523	56
station2	YUV4MPEG	973557114	484974179	50
waterfall	YUV4MPEG	39538244	30113554	76
bach	WAV	1672716	1483903	89
quartet	WAV	2164994	1723231	80
voice	WAV	1958598	1425009	73
enwik8	ASCII	100000000	35103261	35
fork	ASCII	77302	22328	29
artificial	PNM	37748755	15984463	42
big_building	PNM	234317971	223993316	96
big_tree	PNM	166202419	156856380	94
bridge	PNM	66784225	62564885	94
cathedral	PNM	36096019	30279290	84
deer	PNM	64065397	47759885	75
fireworks	PNM	44255251	18402704	42
flower_foveon	PNM	20575315	16452062	80
hdr	PNM	37748755	34307000	91
leaves_iso_200	PNM	36096019	33983617	94
leaves_iso_1600	PNM	36096019	33890963	94
nightshot_iso_100	PNM	44255251	35616865	80
nightshot_iso_1600	PNM	44255251	36075145	82
spider_web	PNM	72726547	65591005	90

Table 5: Kompresja BZip2 - bzip2

Name	Format before	Size before	Size after	ratio [%]
crew	YUV4MPEG	364957238	170340425	47
station2	YUV4MPEG	973557114	409820858	42
waterfall	YUV4MPEG	39538244	26409574	67
bach	WAV	1672716	1407290	84
quartet	WAV	2164994	1664083	77
voice	WAV	1958598	1187571	61
enwik8	ASCII	100000000	29006372	29
fork	ASCII	77302	20488	27
artificial	PNM	37748755	10996136	29
big_building	PNM	234317971	207733063	89
big_tree	PNM	166202419	142958710	86
bridge	PNM	66784225	53886452	81
cathedral	PNM	36096019	25418438	70
deer	PNM	64065397	34425145	54
fireworks	PNM	44255251	13973511	32
flower_foveon	PNM	20575315	14651775	71
hdr	PNM	37748755	28578653	76
leaves_iso_200	PNM	36096019	31521131	87
leaves_iso_1600	PNM	36096019	31752719	88
nightshot_iso_100	PNM	44255251	29601706	67
nightshot_iso_1600	PNM	44255251	30515333	69
spider_web	PNM	72726547	59728101	82

Table 6: Kompresja LZMA - zip

Name	Format before	Size before	Size after	ratio [%]
crew	YUV4MPEG	364957238	179050453	49
station2	YUV4MPEG	973557114	433383702	45
waterfall	YUV4MPEG	39538244	26427150	67
bach	WAV	1672716	1378378	82
quartet	WAV	2164994	1598041	74
voice	WAV	1958598	1202250	61
enwik8	ASCII	100000000	24857881	25
fork	ASCII	77302	21540	28
artificial	PNM	37748755	6742449	18
big_building	PNM	234317971	205600142	88
big_tree	PNM	166202419	140422608	84
bridge	PNM	66784225	53454752	80
cathedral	PNM	36096019	26097758	72
deer	PNM	64065397	32390714	51
fireworks	PNM	44255251	14746233	33
flower_foveon	PNM	20575315	12891952	63
hdr	PNM	37748755	29251806	77
leaves_iso_200	PNM	36096019	31389078	87
leaves_iso_1600	PNM	36096019	31799192	88
nightshot_iso_100	PNM	44255251	30152901	68
nightshot_iso_1600	PNM	44255251	30670207	69
spider_web	PNM	72726547	48773634	67

Table 7: Kodowanie tekstu - porównanie

Name	ratio (Deflate)	ratio (BZip2) [%]	ratio (LZMA) [%]
crew	56	47	49
station2	50	42	45
waterfall	76	67	67
bach	89	84	82
quartet	80	77	74
voice	73	61	61
enwik8	35	29	25
fork	29	27	28
artificial	42	29	18
big_building	96	89	88
big_tree	94	86	84
bridge	94	81	80
cathedral	84	70	72
deer	75	54	51
fireworks	42	32	33
flower_foveon	80	71	63
hdr	91	76	77
leaves_iso_200	94	87	87
leaves_iso_1600	94	88	88
nightshot_iso_100	80	67	68
nightshot_iso_1600	82	69	69
spider_web	90	82	67

Wyniki kompresji są zbliżone dla każdego z trzech algorytmów. Najlepiej poradził sobie kodek LZMA, który ma najniższe lub porównywalne ratio dla każdego typu danych. Najgorzej natomiast zadziałał algorytm Deflate, który dał najmniejsze zmniejszenie nadmiarowości. Duże znaczenie przy kompresji algorytmami ogólnego zastosowania ma typ danych. Najlepiej kompresowane były zwykłe pliki tekstowe zawierające zdania w języku angielskim albo kod źródłowy programów. Pozostałe typy kompresowały się w różnym stopniu, w zależności od ich zapisu binarnego, jednakże widać tendencję dla danego pliku. Jedne z nich miały względnie niskie wartości ratio, inne natomiast względnie wysokie.

6 Kodowanie obrazu

Kompresję bezstratną obrazów wykonano używając dwóch popularnych koderów. Pierwszy z nich jest powszechnie używany w zastosowaniach domowych, a drugi w kodowaniu zdjęć przesyłanych przez internet. Dla każdego kodeka wyłączono dodawanie do pliku metadanych, żeby zniwelować ich wpływ na rozmiar. Dla obydwu koderów została włączona maksymalna kompresja danych, a dla WebP została dodatkowo włączona opcja kompresji bezstratnej, gdyż umożliwia on także kodowanie stratne obrazów. Danymi wejściowymi były różnego rodzaju i wielkości zdjęcia w kodowaniu bezstratnym Netpbm o rozszerzeniu PNM, gdyż ważne było użycie danych poddanych jak najmniejszej obróbce aby nie zafałszować wyników kompresji. Zdjęcia te były zapisane w 16 bitowej przestrzeni kolorów RGB. Do kompresji został użyty program GIMP 2.10.8.

7 Wyniki kodowania obrazu

Table 8: Kompresja PNG

Name	Format before	Size before	Size after	ratio [%]
artificial	PNM	37748755	7497925	20
big_building	PNM	234317971	199358332	85
big_tree	PNM	166202419	144518168	87
bridge	PNM	66784225	59012677	88
cathedral	PNM	36096019	29617638	82
deer	PNM	64065397	58241967	91
fireworks	PNM	44255251	21339086	48
flower_foveon	PNM	20575315	15004662	73
hdr	PNM	37748755	29597826	78
leaves_iso_200	PNM	36096019	30578665	85
leaves_iso_1600	PNM	36096019	31585619	88
nightshot_iso_100	PNM	44255251	32520846	73
nightshot_iso_1600	PNM	44255251	36996116	84
spider_web	PNM	72726547	53482310	74

Table 9: Kompresja WebP

Name	Format before	Size before	Size after	ratio [%]
artificial	PNM	37748755	1017688	3
big_building	PNM	234317971	53920370	23
big_tree	PNM	166202419	45931996	28
bridge	PNM	66784225	19396168	29
cathedral	PNM	36096019	8421942	23
deer	PNM	64065397	20772478	32
fireworks	PNM	44255251	4795092	11
flower_foveon	PNM	20575315	2449486	12
hdr	PNM	37748755	5847608	15
leaves_iso_200	PNM	36096019	8479040	23
leaves_iso_1600	PNM	36096019	10305046	29
nightshot_iso_100	PNM	44255251	6423444	15
nightshot_iso_1600	PNM	44255251	11706714	26
spider_web	PNM	72726547	8426556	12

Table 10: Kodowanie obrazu - porównanie

Name	ratio (PNG)	ratio (WebP) [%]
artificial	20	3
big_building	85	23
big_tree	87	28
bridge	88	29
cathedral	82	23
deer	91	32
fireworks	48	11
flower_foveon	73	12
hdr	78	15
leaves_iso_200	85	23
leaves_iso_1600	88	29
nightshot_iso_100	73	15
nightshot_iso_1600	84	26
spider_web	74	12

Kompresja obrazów dała bardzo ciekawe wyniki. Najlepiej z nią poradził sobie kodek WebP, który potrafił dokonać kompresji bezstratnej od 3 do 6 razy lepiej niż popularny PNG. Dla pierwszego obrazu (artificial) ratio wyniosło tylko 3% dla WebP, co jest najlepszym wynikiem z zestawienia. Prawdopodobnie wynika to z typu tego obrazu, który został wygenerowany na komputerze oraz zawiera duże powierzchnie jednolitego koloru. Dla pozostałych obrazów kodek PNG pozwalał na obniżenie wielkości o około 10-30%, a kodek WebP o 70-90%. Ciekawa różnica występuje pomiędzy bardzo podobnymi obrazami z innym współczynnikiem ISO odpowiadającym za czułość aparatu na światło. Przy wyższym wskaźniku kompresja dawała gorsze wyniki.

8 Kodowanie wideo

Kompresję bezstratną filmów wykonano używając trzech kodeków wymienionych w tabeli poniżej.

kodek	kontener	głębia bitowa
H.264	MPEG4	8 bit
H.265	FFMPEG	8 bit
FFV1	FFMPEG	10 bit

Dla każdego kodeka wybrano kompresję bezstratną na najwyższych ustawieniach kompresji. Dane wejściowe były w formacie YUV4MPEG, który pozwala na zapisanie filmu jako krótkiego headera z podstawowymi informacjami oraz nielimitowanej ilości uszeregowanych zdjęć, w tym przypadku skompresowanych bezstratnie. Zestaw testowy zawiera trzy filmy, których wielkość pozwala na wykonanie kompresji w ciągu kilku minut na komputerach osobistych, a jednocześnie reprezentują dużą grupę istniejących nagrań. Dłuższe filmy musiałby zostać odrzucone ze względu na ich dużą wielkość (sięgającą kilkudziesięciu GB) oraz czas ich kompresji, który mógłby zająć ponad dzień. Do kompresji został użyty program VirtualDub2.

9 Wyniki kodowania wideo

Table 12: Kompresja x264 lossless - MPEG4

Name	Format before	Size before	Size after	ratio [%]
crew	YUV4MPEG	364957238	146310248	40
station2	YUV4MPEG	973557114	361046080	37
waterfall	YUV4MPEG	39538244	14465242	37

Table 13: Kompresja x265 lossless - FFMPEG

Name	Format before	Size before	Size after	ratio [%]
crew	YUV4MPEG	364957238	167599010	46
station2	YUV4MPEG	973557114	372919858	38
waterfall	YUV4MPEG	39538244	13732228	35

Table 14: Kompresja ffv1 - FFMPEG

Name	Format before	Size before	Size after	ratio [%]
crew	YUV4MPEG	364957238	192322936	53
station2	YUV4MPEG	973557114	485438682	50
waterfall	YUV4MPEG	39538244	33615372	85

Table 15: Kodowanie filmu - porównanie

Name	ratio (x264)	ratio (x265) [%]	ratio (ffv1) [%]
crew	40	46	53
station2	37	38	50
waterfall	37	35	85

Wszystkie kodeki całkiem dobrze poradziły sobie z kodowanie filmów, które pozwoliło na zmniejszenie nadmiarowości o około 50-60%. Na badanych źródłach danych najlepiej poradził sobie kodek x264 który przeprowadził kompresję o 5-10 punktów procentowych lepiej od konkurencji. O ile w kodowaniu stratnym wg innych badań oraz doświadczeń eksperymentatora kodek x265 dużo lepiej sobie radzi z kompresją to w przypadku kompresji bezstratnej wypadł porównywalnie jak jego poprzednik, czyli kodek x264. Natomiast kodek ffv1 nie poradził sobie z filmem “waterfall”, gdyż zredukował jego wielkość jedynie o 15%.

10 Kodowanie dźwięku

Do kodowania dźwięku użyto dwóch kodeków: FLAC oraz ALAC. Obydwa kompresowały dane z ustawieniami najlepszej kompresji. Pliki do formatu FLAC były kompresowane za pomocą programu Audacity, a format ALAC był uzyskany przy pomocy konwertera online. Trzy próbki dźwięku zostały pobrane w formacie bezstratnym PCM, który był najbardziej zbliżony do rzeczywistych niezmodyfikowanych danych. Dźwięk miał częstotliwość 44.1 kHz z częstotliwością próbkowania 16 bitów.

11 Wyniki kodowania dźwięku

Table 16: Kompresja FLAC

Name	Format before	Size before	Size after	ratio [%]
bach	WAV	1672716	743476	44
quartet	WAV	2164994	945611	44
voice	WAV	1958598	839281	43

Table 17: Kompresja ALAC

Name	Format before	Size before	Size after	ratio [%]
bach	WAV	1672716	755978	45
quartet	WAV	2164994	946188	44
voice	WAV	1958598	817552	42

Table 18: Kodowanie dźwięku - porównanie

Name	ratio (FLAC)	ratio (ALAC) [%]
bach	44	45
quartet	44	44
voice	43	42

Oba kodeki zredukowały wielkość plików o około 60% co jest dobrym wynikiem jak na kodek bezstratny. Ponieważ dźwięk zazwyczaj jest kodowany metodą stratną to kodery bezstratne nie są często używane - jedynie w sytuacjach kiedy ważne jest dokładne odwzorowanie dźwięku wejściowego.

12 Podsumowanie wyników

Wszystkie kodery pozwoliły zmniejszyć nadmiarowość od 10 do 90 procent w zależności od danych wejściowych. Większość z nich pozwala na redukcję wielkości danego pliku w podobnym zakresie co konkurencyjne kodeki, jednak było co do tego kilka wyjątków. W kodowaniu dźwięku oraz video żaden z koderów nie wyróżnia się znacząco na tle konkurencji. Natomiast w kodowaniu obrazów wygranym jest kodek WebP który zdeklasował popularnie używaną konkurencję w tym zakresie.

Ciekawe jest natomiast porównanie działania koderów dedykowanych dla danego typu danych i koderów ogólnego użycia. Potrafiły one z łatwością kompresować tekst, jednakże dały także dobre wyniki kompresji wszystkich innych typów danych. To jak dobrze zostały one skompresowane prawdopodobnie wynika z charakteru kodera źródłowego danych, gdyż przy ich zastosowaniu ważna jest reprezentacja bitowa danych.

Podsumowując kodowanie bezstratne pozwala na znaczące zmniejszenie nadmiarowości danych przy jednoczesnym zagwarantowaniu identyczności z danymi źródłowymi, co często jest kluczowe w dalszym przetwarzaniu tych danych.

13 Bibliografia

- 7-zip: https://7-zip.org.pl/
- GIMP: https://www.gimp.org/
- Audacity: https://audacity.pl/
- ALAC online converter: https://www.onlineconverter.com/alac
- VirtualDub2: https://sourceforge.net/p/vdfiltermod/wiki/Home/