Raport 6

# Dane testowe do kompresji

W tabelach zamieszczono jedynie pierwsze 5 elementów, aby nie zajmować zbyt dużo miejsca.

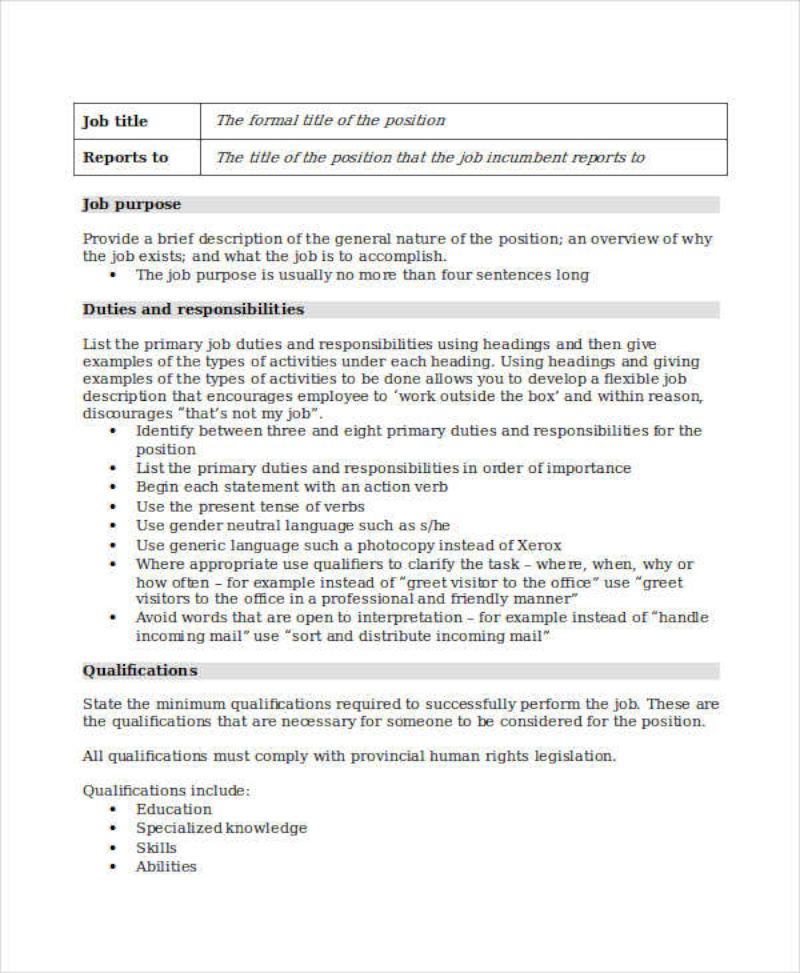
## Dane testowe do kompresji, algorytm RLE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dane oryginalne | Dane skompresowane | Dane zdekompresowane | Rozmiar oryginalny | Rozmiar skompresowany | Czy oryginał = dekompresja |
| [1, 1, 1, 1, 2]... | [4, 1, 1, 2, 4]... | [1, 1, 1, 1, 2]... | 112 | 156 | True |
| [1, 2, 3, 1, 2]... | [1, 1, 1, 2, 1]... | [1, 2, 3, 1, 2]... | 72 | 220 | True |
| [5, 1, 5, 1, 5]... | [1, 5, 1, 1, 1]... | [5, 1, 5, 1, 5]... | 104 | 220 | True |
| [-1, -1, -1, -5, -5]... | [3, -1, 2, -5, 1]... | [-1, -1, -1, -5, -5]... | 96 | 204 | True |
| [0.0, 0.0, 0.0, 0.0, 0.0]... | [520, 0] | [0, 0, 0, 0, 0]... | 4160 | 128 | True |
| [0, 1, 2, 3, 4]... | [1, 0, 1, 1, 1]... | [0, 1, 2, 3, 4]... | 4168 | 8412 | True |
| [1.0, 0.0, 0.0, 0.0, 0.0]... | [1, 1, 7, 0, 1]... | [1, 0, 0, 0, 0]... | 392 | 292 | True |
| [1.0, 1.0, 1.0, 0.0, 0.0]... | [3, 1, 21, 0, 3]... | [1, 1, 1, 0, 0]... | 1176 | 328 | True |
| [1.0, 1.0, 1.0, 1.0, 1.0]... | [10, 1] | [1, 1, 1, 1, 1]... | 80 | 168 | True |

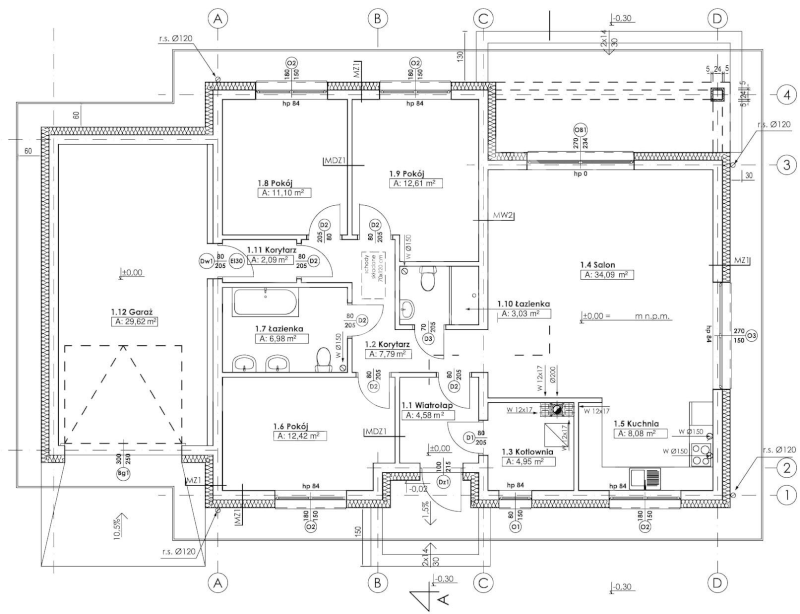
## Dane testowe do kompresji, algorytm ByteRun

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dane oryginalne | Dane skompresowane | Dane zdekompresowane | Rozmiar oryginalny | Rozmiar skompresowany | Czy oryginał = dekompresja |
| [1, 1, 1, 1, 2]... | [-3, 1, 0, 2, -3]... | [1, 1, 1, 1, 2]... | 112 | 156 | True |
| [1, 2, 3, 1, 2]... | [8, 1, 2, 3, 1]... | [1, 2, 3, 1, 2]... | 72 | 156 | True |
| [5, 1, 5, 1, 5]... | [3, 5, 1, 5, 1]... | [5, 1, 5, 1, 5]... | 104 | 196 | True |
| [-1, -1, -1, -5, -5]... | [-2, -1, -1, -5, 3]... | [-1, -1, -1, -5, -5]... | 96 | 180 | True |
| [0.0, 0.0, 0.0, 0.0, 0.0]... | [-127, 0, -127, 0, -127]... | [0, 0, 0, 0, 0]... | 4160 | 192 | True |
| [0, 1, 2, 3, 4]... | [127, 0, 1, 2, 3]... | [0, 1, 2, 3, 4]... | 4168 | 4284 | True |
| [1.0, 0.0, 0.0, 0.0, 0.0]... | [0, 1, -6, 0, 0]... | [1, 0, 0, 0, 0]... | 392 | 292 | True |
| [1.0, 1.0, 1.0, 0.0, 0.0]... | [-2, 1, -20, 0, -2]... | [1, 1, 1, 0, 0]... | 1176 | 328 | True |
| [1.0, 1.0, 1.0, 1.0, 1.0]... | [-9, 1] | [1, 1, 1, 1, 1]... | 80 | 168 | True |

# Obrazy do kompresji







|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Nr zdjęcia | Rozmiar oryginalny | Rozmiar skompresowany RLE | Rozmiar skompresowany ByteRun | Stopień kompresji RLE | Stopień kompresji ByteRun | Czy oryginał = dekompresja RLE | Czy oryginał = dekompresja ByteRun |
| Obraz 1 | 18681600 | 11309668 | 6735500 | 1.6518256769341062 | 2.7736025536337316 | True | True |
| Obraz 2 | 15364800 | 30006132 | 15563836 | 0.5120553358893443 | 0.9872116360002765 | True | True |
| Obraz 3 | 11788800 | 1943604 | 2032004 | 6.065433082047577 | 5.801563382749246 | True | True |

Skompresowane wektory są przechowywane w klasie CompressedData, która zawiera: Skompresowaną tablice, informacje o oryginalnym kształcie I informacje o rozmiarze całego obiektu, czyli rozmiaru tablicy + rozmiaru informacji o kształcie.

Porównanie macierzy przed I po kompresji w celu sprawdzenia, czy została uzyskana oryginalna macierz, wykonywane jest poprzez funkcję

np.array\_equal()

# Obserwacje I wnioski

Obrazy proste, o ograniczonej liczbie kolorów, w których te same barwy często występują obok siebie, są skutecznie kompresowane przy użyciu algorytmów opracowanych podczas zajęć laboratoryjnych. Jednak w przypadku zastosowanego kolorowego obrazu, rozmiar pliku po kompresji okazał się większy niż rozmiar oryginału — i to dla obu badanych algorytmów. Warto też dodać, że dla niektórych danych testowych sporą różnicę zrobiło przechowywanie informacji o oryginalnym kształcie danych w tej samej klasie. Choć sam rozmiar danych po kompresji jest mniejszy, tak rozmiar całkowitej paczki danych spowodował że dane testowe zajmują więcej miejsca niż zajmowały przed kompresją.

Wynika z tego, że zarówno algorytm RLE, jak i ByteRun, sprawdzają się dobrze w przypadku danych, w których często powtarzają się identyczne elementy znajdujące się w bezpośrednim sąsiedztwie. Gdy natomiast dane są zróżnicowane lub mimo niewielkiej różnorodności te same wartości nie pojawiają się obok siebie, algorytmy te nie tylko zawodzą w zmniejszaniu rozmiaru, ale wręcz mogą prowadzić do jego zwiększenia.