

Instrukcja 05- Kompresja bezstratna- RLE, drzewo czwórkowe

3.1 Proszę poświęcić paragraf sprawozdania i opisać sposób wykorzystania pamięci przez wasze implementacje. Gdzie przechowywanych jest rozmiar obrazu w każdym z skompresowanych plików oraz jaką strukturą w pamięci jest drzewo.

a) RLE

Pierwsze dwa elementy arraya 1D wyniku kompresji to wymiary obrazu, przez co przy dekompresji, pętla ma odpowiednio przesunięte wartości indeksów. W retrospekcji, praktyczniej byłoby umieścić rozmiary na końcu.

b) QuadTree

Tutaj podobnie jak w RLE, wielkość obrazu znajduje się na początku obiektu skompresowanego obrazu, z tą różnicą, że obiekt to [obraz.shape, dane_drzewa], więc na początku dekompresji, funkcja pobiera wartości z pierwszego elementu, a potem dobiera się do drzewa.

Samo drzewo zbudowane jest na klasie, którą nazwałem branch. Zawiera ona zmienną typu bool, która pokazuje czy jest to liść. Każdy branch zawiera swoje dzieci (od 1 do 4), a jeśli nie ma dzieci, czyli jest liściem, to zawiera dane o kolorze. Każdy obiekt branch ma też współrzędne wierzchołka oraz długości boków, które są używane przy tworzeniu drzewa, oraz przy nawigacji podczas dekompresji.

3.2 Dla każdego z przygotowanych obrazów testowych przeprowadzić kompresję i dekompresję, każda z metod. Udowodnić poprawność (identyczność) danych po dekompresji z danymi źródłowymi. Sprawdzić skuteczność kompresji - zarówno jako jej stopień jak i jaki % oryginalnego obrazu stanowi obraz po kompresji.

a) rysunek techniczny

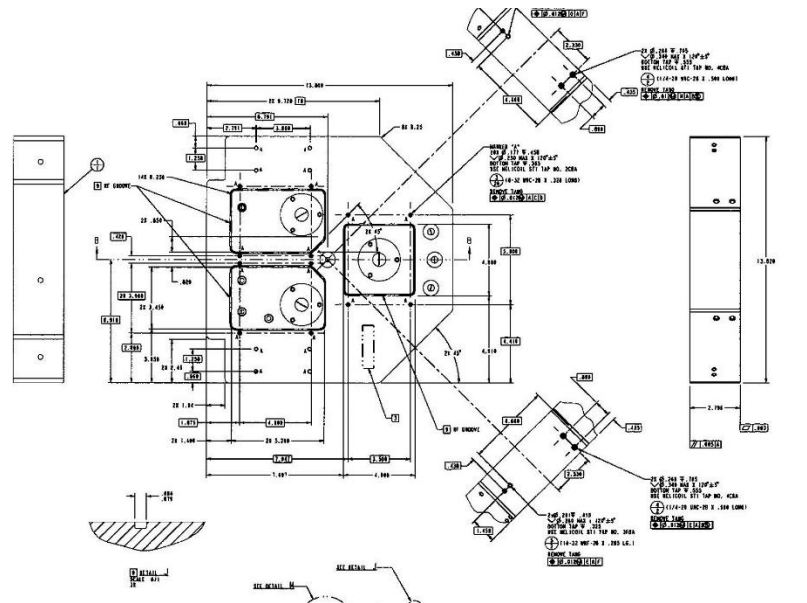


Figure 1 rysunek techniczny - oryginał

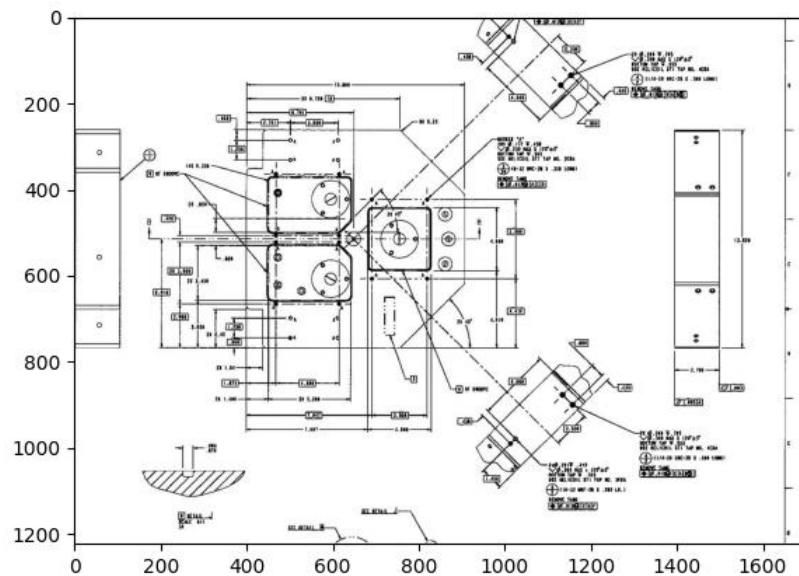


Figure 2 rysunek techniczny - RLE

```
(1224, 1696, 3)
dekodek
12455560
13733016
stopien kompresji: 0.9069792098108674 , 110.25611052413541 %
Obraz oryginalny i wynik dekompresji maja te same wartosci.
```

Figure 3 rysunek techniczny - RLE, wyniki

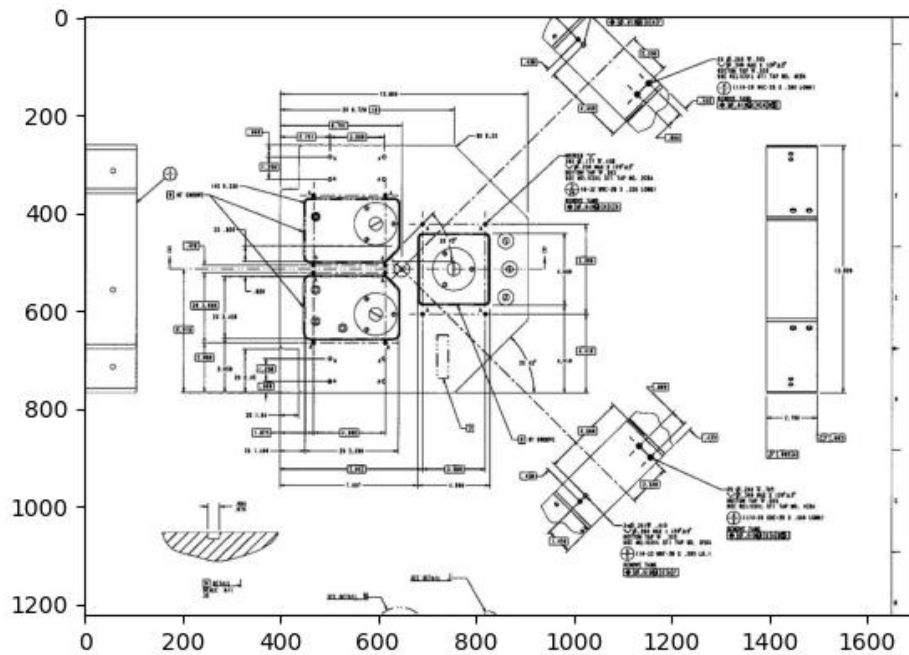


Figure 4 rysunek techniczny - Quadtree

```
(1224, 1696, 3)
dekodek
12455560
1141
stopien kompresji: 10916.354075372481 , 0.009160567650109669 %
Obraz oryginalny i wynik dekompresji maja te same wartosci.
```

Figure 5 rysunek techniczny - Quadtree, wyniki

ZARZĄDZENIE NR 35
Rektora Zachodniopomorskiego Uniwersytetu Technologicznego w Szczecinie
z dnia 22 maja 2018 r.
w sprawie procedury postępowania przy ubieganiu się o środki finansowe oraz przy realizacji i rozliczania projektów finansowanych z funduszy zewnętrznych

Na podstawie art. 66 ust. 2 ustawy Prawo o szkolnictwie wyższym (tekst jedn. Dz. U. z 2017 r. poz. 2183, z późn. zm.) zarządza się, co następuje:

§ 1.

Wprowadza się Procedurę postępowania przy ubieganiu się o środki finansowe oraz przy realizacji i rozliczaniu projektów finansowanych z funduszy zewnętrznych, stanowiącą załącznik do niniejszego zarządzenia.

§ 2.

Tracą moc:

- 1) zarządzenie nr 47 Rektora ZUT z dnia 4 lutego 2009 r. w sprawie wprowadzenia „Zasad dotyczących trybu postępowania w procesie realizacji projektów finansowanych ze środków funduszy strukturalnych oraz kompetencje i obowiązki osób i komórek organizacyjnych” wraz z późniejszymi zmianami wprowadzonymi: zarządzeniem nr 93 Rektora ZUT z dnia 10 czerwca 2009 r. zarządzeniem nr 63 Rektora ZUT z dnia 15 lipca 2010 r., zarządzeniem nr 23 Rektora ZUT z dnia 21 kwietnia 2017 r.;
- 2) zarządzenie nr 86 Rektora ZUT z dnia 25 maja 2009 r. w sprawie wprowadzenia „Procedury postępowania przy ubieganiu się o środki z funduszy strukturalnych oraz sposób realizacji i rozliczania projektów” wraz z późniejszymi zmianami wprowadzonymi: zarządzeniem nr 62 Rektora ZUT z dnia 15 lipca 2010 r., zarządzeniem nr 10 Rektora ZUT z dnia 17 lutego 2016 r.;
- 3) zarządzenie nr 3 Rektora ZUT z dnia 15 stycznia 2015 r. w sprawie wdrożenia procedury postępowania przy ubieganiu się o środki finansowe oraz sposobu realizacji i rozliczania projektów finansowanych ze środków będących w dyspozycji ministra właściwego ds. nauki”.

§ 3.

Zarządzenie wchodzi w życie z dniem podpisania, z mocą obowiązującą od dnia 1 czerwca 2018 r.

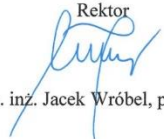
Rektor

dr hab. inż. Jacek Wróbel, prof. nadzw.

Figure 6 skan dokumentu - oryginał

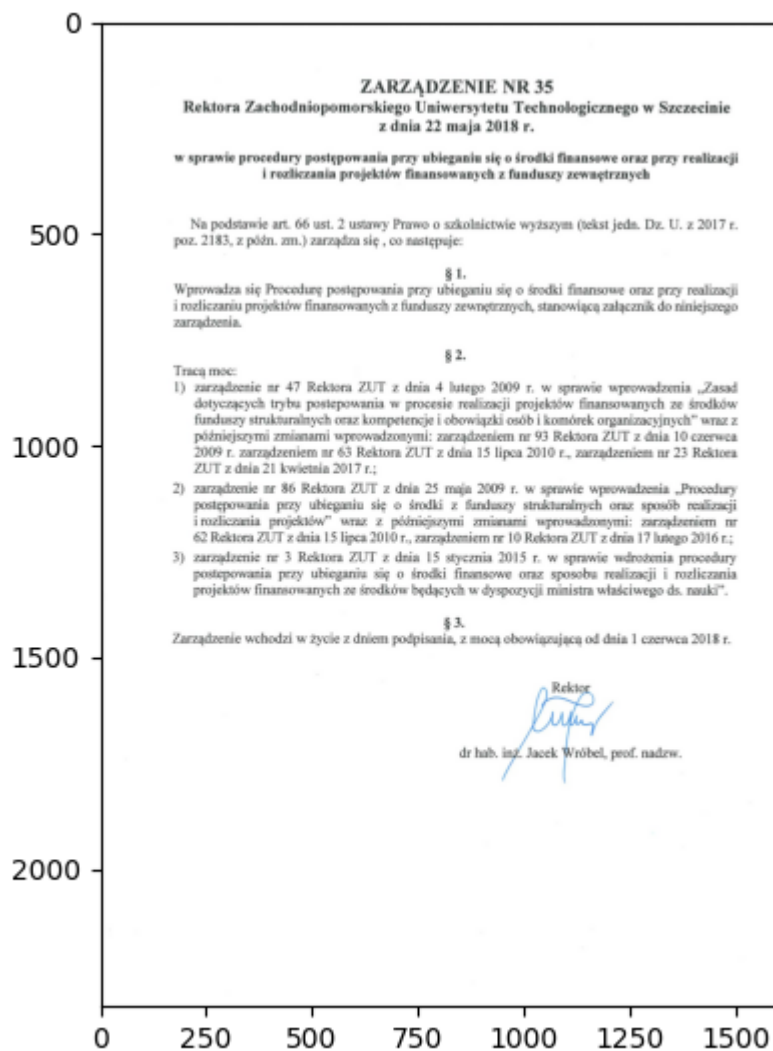


Figure 7 skan dokumentu - RLE

```
(2323, 1606, 3)
dekoeder
22384564
20800824
stopien kompresji: 1.076138329904623 , 92.92485661101105 %
Obraz oryginalny i wynik dekompresji maja te same wartosci.
```

Figure 8 skan dokumentu - RLE, wyniki



Figure 9 skan dokumentu – Quadtree

```
(2323, 1606, 3)
dekoader
22384564
1141
stopien kompresji: 19618.373356704644 , 0.005097262560039142 %
Obraz oryginalny i wynik dekompresji maja te same wartości.
```

Figure 10 skan dokumentu - Quatree, wyniki

c) kolorowe zdjęcie



Figure 11 kolorowe zdjęcie - oryginał

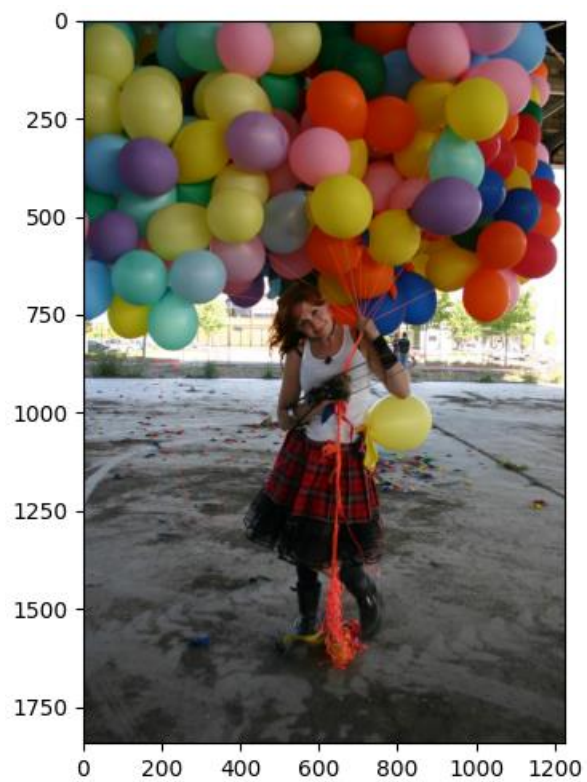


Figure 12 kolorowe zdjęcie - RLE


```
(1843, 1228, 3)
deko
13579360
53794840
stopien kompresji: 0.25242867159749893 , 396.15151229513026 %
Obraz oryginalny i wynik dekompresji maja te same wartosci.
```

Figure 13 kolorowe zdjęcie - RLE, wyniki



Figure 14 kolorowe zdjęcie – QuadTree


```
(1843, 1228, 3)
dekoder
13579360
1141
stopien kompresji: 11901.27957931639 , 0.008402457847792533 %
Obraz oryginalny i wynik dekompresji maja te same wartosci.
```

Figure 15 kolorowe zdjęcie - QuadTree, wyniki

Wnioski:

Ujemna kompresja w RLE* wynika z typu obrazów, chociaż przy rysunku technicznym spodziewałem się przynajmniej zauważalnej kompresji. Format obrazów jpg, który bardzo często ma masę punktowych przebarwień, wpływa negatywnie na obie formy kompresji bezstratnej, które polegają w tym przypadku na identycznych barwach sąsiednich pikseli. Nałożenie tolerancji barwy i przejście na kompresję stratną, prawdopodobnie poprawiłoby wyniki.

Przeprowadziłem test RLE z dwukolorowym obrazem, podzielonym idealnie równoległą linią poziomą:



Figure 16 obraz użyty do sprawdzenia RLE - flaga Polski

Obraz utworzyłem sam i zapisałem do formatu jpg, tego samego w którym były poprzednie obrazy, co wpłynęło na zmiany w pikselach na granicy kolorów:



Figure 17 duże przybliżenie granicy kolorów obrazu testowego

Wynik kompresji RLE poprawił się drastycznie, dowodzi on, zaproponowanej przeze mnie hipotezy o jakości, typie i formacie użytych obrazów.

```
(1224, 1696, 3)
dekomder
12455560
54744
stopien kompresji: 227.52374689463684 , 0.4395145621714319 %
Obraz oryginalny i wynik dekompresji maja te same wartości.
```

Figure 18 obraz testowy - RLE, wyniki

Każdy obraz po kompresji QuadTree jest tego samego rozmiaru**, 1141 co niemal gwarantuje błędne pomiary, pomimo użycia funkcji zaproponowanej w instrukcji. Obstawiam, że funkcja liczy tylko rozmiar klasy pierwszej, pnia drzewa, a ignoruje dzieci, i tym samym pozostałe dane w liściach.

*Po zmianie obrazu skanu dokumentu z dowodu osobistego na stronę z zarządzenia rektora zapisaną jako obraz w formacie jpg, kompresja obrazu wyszła mniejsza, niż oryginał!

**Po zakomentowaniu dwóch linijek w funkcji `get_size`:

```
65         seen.add(obj_id)
66     #     if isinstance(obj, np.ndarray):
67     #         size += obj.nbytes
68     if isinstance(obj, dict):
```

Figure 19 zedytowany fragment `get_dize`

na rdp zutu, python pokazywał inne, zdecydowanie większe wartości po kompresji QuadTree, jednak na moim komputerze rozmiary pozostały 1141! Zostawiłem zatem te wyniki jako nauka na przyszłość, a większy rozmiar po kompresji QuadTree zapewne wynika z przechowywania danych w zagnieżdżonych klasach, co jest bardzo mało wydajne, w porównaniu do np. przechowywania w arrayu podobnie do RLE.