# **LAPORAN PRAKTIKUM 11**

***IMAGE COMPRESSION***

****

**Oleh:**

|  |  |  |
| --- | --- | --- |
| **ADINDA PUTRI N.R** | **-** | **2005001** |
| **BIDIN WAHYUDIN S.P** | **-** | **2005005** |
| **LUSIYANA** | **-** | **2005014** |
| **M. RIZKY PRATAMA** | **-** | **2005018** |

**PROGRAM STUDI REKAYASA PERANGKAT LUNAK**

**JURUSAN TEKNIK INFORMATIKA**

**POLITEKNIK NEGERI INDRAMAYU**

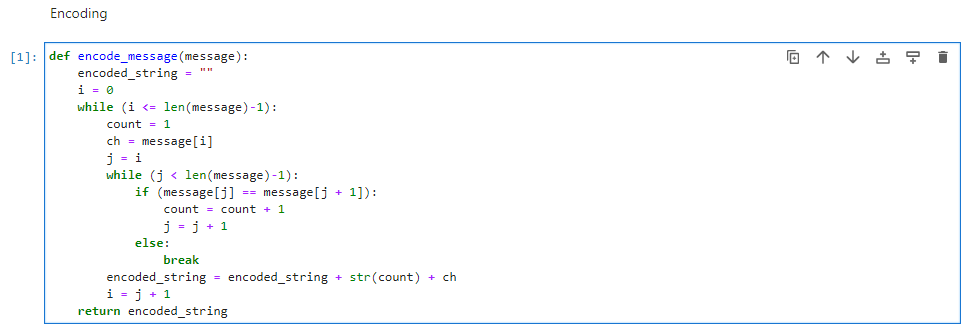
**MEI 2023**

**Deskripsi**

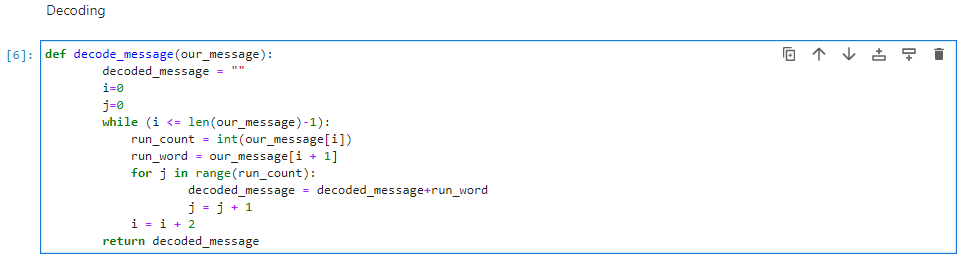
Kompresi data adalah bagian yang sangat penting dari dunia digital, di mana kita memiliki banyak file dengan ukuran besar. Banyak faktor dari hal tersebut diantaranya *smartphone* memiliki kualitas kamera yang lebih baik dan gambar yang diambil darinya juga membutuhkan lebih banyak penyimpanan, dengan kombinasi piksel yang lebih kompleks, lebih banyak penyimpanan yang dibutuhkan. Kompresi dengan format data tertentu seperti JPEG dan PNG, memungkinkan sebagian data hilang. Di sisi lain informasi detail dari gambar dibutuhkan untuk memproses gambar tersebut. Maka kita perlu menggunakan metode konversi *lossless* yang artinya tidak boleh ada kehilangan data.

**Praktikum**

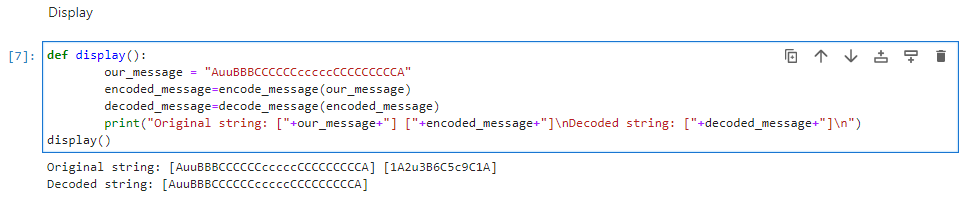
1. **RLE (*Run Length Encoding*) sederhana untuk data**
   1. ***Encoding***



* Fungsi "encode\_message" didefinisikan dengan satu parameter "message".
* Variabel "encoded\_string" diinisialisasi dengan nilai string kosong. Variabel "i" diinisialisasi dengan nilai 0.
* Loop while dimulai, yang akan berjalan selama nilai "i" kurang dari atau sama dengan panjang pesan dikurangi 1.
* Variabel "count" diinisialisasi dengan nilai 1. Variabel "ch" diinisialisasi dengan karakter pada indeks "i" dalam pesan.
* Variabel "j" diinisialisasi dengan nilai "i". Loop while kedua dimulai, yang akan berjalan selama "j" kurang dari panjang pesan dikurangi 1.
* Dalam loop while kedua, cek apakah karakter pada indeks "j" sama dengan karakter pada indeks "j + 1" dalam pesan. Jika ya, tambahkan 1 ke variabel "count" dan tambahkan 1 ke "j". Jika tidak, keluar dari loop while kedua.
* Tambahkan hasil encoding karakter dengan "jumlah kemunculan + karakter" ke variabel "encoded\_string".
* Atur nilai "i" menjadi "j + 1" untuk memulai pencarian karakter baru pada pesan.
* Setelah loop while pertama selesai, kembalikan nilai "encoded\_string".
* Fungsi selesai.
  1. ***Decoding***



* Fungsi "decode\_message" didefinisikan dengan satu parameter "our\_message".
* Variabel "decoded\_message" diinisialisasi dengan nilai string kosong. Variabel "i" dan "j" diinisialisasi dengan nilai 0.
* Loop while dimulai, yang akan berjalan selama nilai "i" kurang dari atau sama dengan panjang pesan yang dienkripsi dikurangi 1.
* Variabel "run\_count" diinisialisasi dengan nilai integer yang diambil dari karakter pada indeks "i" dalam pesan.
* Variabel "run\_word" diinisialisasi dengan karakter pada indeks "i + 1" dalam pesan.
* Loop for dimulai, yang akan berjalan sebanyak "run\_count" kali.
* Dalam loop for, tambahkan karakter "run\_word" ke variabel "decoded\_message".
* Atur nilai "j" menjadi "j + 1".
* Atur nilai "i" menjadi "i + 2" untuk memulai pencarian karakter enkripsi baru pada pesan.
* Setelah loop while pertama selesai, kembalikan nilai "decoded\_message".
* Fungsi selesai.
  1. ***Display***

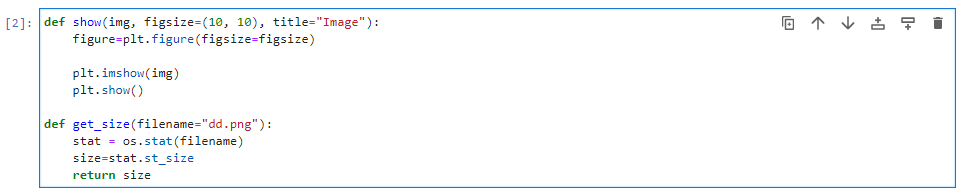


* Fungsi "display" didefinisikan tanpa parameter.
* Variabel "our\_message" diinisialisasi dengan nilai string "AuuBBBCCCCCCcccccCCCCCCCCCA".
* Variabel "encoded\_message" diinisialisasi dengan memanggil fungsi "encode\_message" dengan argumen "our\_message".
* Variabel "decoded\_message" diinisialisasi dengan memanggil fungsi "decode\_message" dengan argumen "encoded\_message".
* Sebuah pesan teks yang dihasilkan akan dicetak ke konsol dengan memformat string. Pertama, pesan teks "Original string:" diikuti oleh nilai variabel "our\_message" dan "encoded\_message" dipisahkan oleh tanda koma. Kemudian, pesan teks "Decoded string:" diikuti oleh nilai variabel "decoded\_message".
* Fungsi "display" dipanggil di akhir program untuk menampilkan hasilnya ketika program dijalankan.

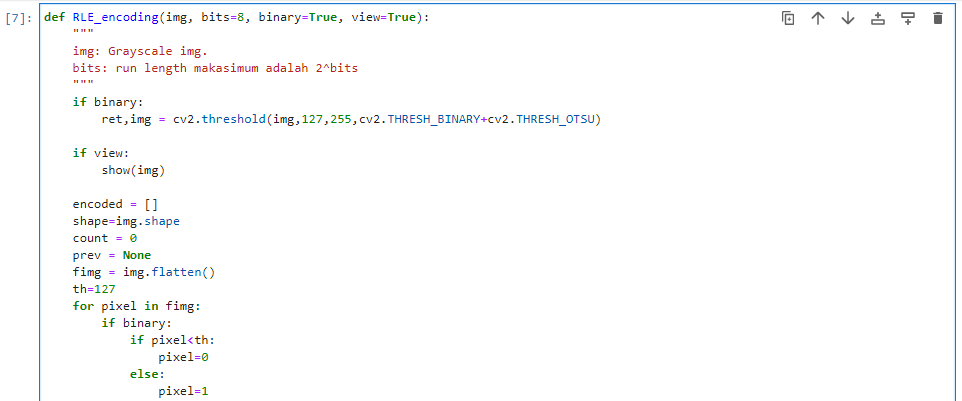
1. **RLE (*Run Length Encoding*) Gambar Sederhana**
   1. ***Import Library***

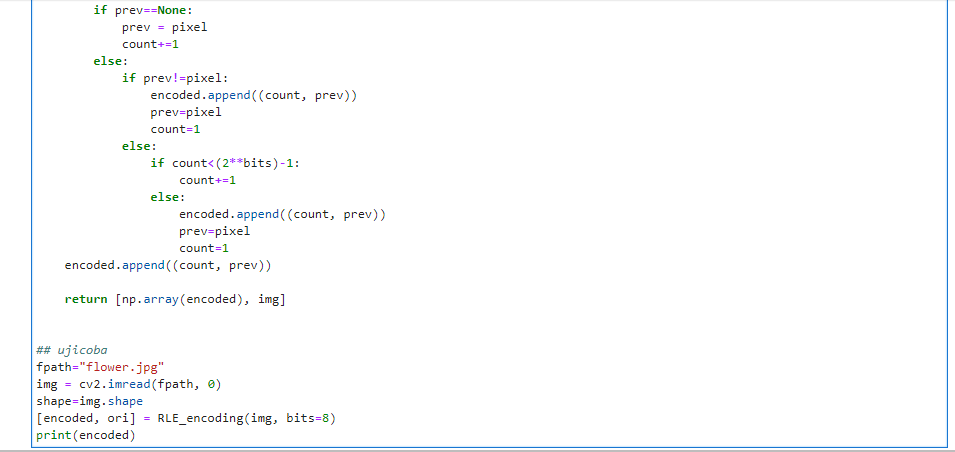


* *Import library* yang dibutuhkan yaitu **numpy, cv2, matplotlib.pyplot,** dan **os**
  1. **Fungsi *Basic***

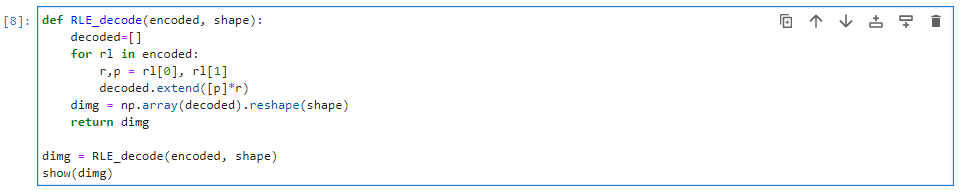


* Fungsi "show" didefinisikan dengan dua parameter: "img" dan "figsize" (dengan nilai default (10, 10)) dan "title" (dengan nilai default "Image").
* Variabel "figure" diinisialisasi dengan objek Figure dari modul matplotlib.pyplot dengan ukuran yang diberikan dalam parameter "figsize".
* Gambar atau array citra yang diberikan sebagai argumen "img" diplot menggunakan fungsi "imshow" dari modul matplotlib.pyplot.
* Fungsi "show" akan menampilkan plot citra yang dihasilkan menggunakan fungsi "show" dari modul matplotlib.pyplot.
* Fungsi "get\_size" didefinisikan dengan satu parameter "filename" (dengan nilai default "dd.png").
* Variabel "stat" diinisialisasi dengan nilai status berkas menggunakan fungsi "os.stat" dari modul os, dan diberikan parameter "filename".
* Variabel "size" diinisialisasi dengan nilai ukuran file yang didapat dari properti "st\_size" pada objek "stat".
* Nilai "size" dikembalikan oleh fungsi "get\_size".
  1. **Encoding**





* Fungsi "RLE\_encoding" didefinisikan dengan tiga parameter: "img" (grayscale image), "bits" (maksimum run length adalah 2^bits), "binary" (dengan nilai default True) dan "view" (dengan nilai default True).
* Jika parameter "binary" adalah True, maka citra grayscale "img" akan diterapkan thresholding menggunakan fungsi "cv2.threshold" dari modul cv2, dengan nilai ambang 127, dan nilai piksel yang di atas nilai ambang akan dijadikan 255, sedangkan piksel di bawah ambang akan dijadikan 0.
* Jika parameter "view" adalah True, maka citra hasil thresholding akan ditampilkan menggunakan fungsi "show".
* Variabel "encoded" diinisialisasi sebagai list kosong yang akan menampung hasil RLE encoding.
* Variabel "shape" diinisialisasi dengan ukuran citra "img".
* Variabel "count" diinisialisasi dengan nilai 0, dan variabel "prev" diinisialisasi dengan nilai None.
* Variabel "fimg" diinisialisasi sebagai flatten array dari citra "img".
* Jika parameter "binary" adalah True, maka setiap piksel dalam "fimg" yang memiliki nilai di bawah 127 akan dijadikan 0, sedangkan piksel yang memiliki nilai di atas atau sama dengan 127 akan dijadikan 1.
* Jika nilai "prev" adalah None, maka variabel "prev" diisi dengan nilai piksel saat ini, dan variabel "count" diincrement dengan nilai 1.
* Jika nilai "prev" tidak sama dengan nilai piksel saat ini, maka tuple (count, prev) akan ditambahkan ke dalam list "encoded", dan nilai "prev" akan diisi dengan nilai piksel saat ini, dan variabel "count" diisi dengan nilai 1.
* Jika nilai "prev" sama dengan nilai piksel saat ini, maka variabel "count" akan diincrement dengan nilai 1. Jika nilai "count" sudah mencapai maksimum run length (2^bits)-1, maka tuple (count, prev) akan ditambahkan ke dalam list "encoded", dan nilai "prev" diisi dengan nilai piksel saat ini, dan variabel "count" diisi dengan nilai 1.
* Tuple terakhir (count, prev) juga ditambahkan ke dalam list "encoded".
* Fungsi "RLE\_encoding" mengembalikan list "encoded" dan citra "img".
* Program ini membaca citra "flower.jpg" dengan menggunakan fungsi "cv2.imread" dari modul cv2 dan mengkonversikannya menjadi citra grayscale. Ukuran citra "img" disimpan dalam variabel "shape".
* Fungsi "RLE\_encoding" dipanggil dengan parameter "bits" bernilai 8. Hasil RLE encoding disimpan dalam variabel "encoded".
  1. **Decoding**



* def RLE\_decode(encoded, shape): : Ini adalah definisi fungsi RLE\_decode yang menerima dua parameter, yaitu hasil encoding RLE encoded dan bentuk asli gambar shape.
* decoded=[] : Inisialisasi variabel decoded sebagai list kosong, yang akan menyimpan hasil decoding.
* for rl in encoded: : Perulangan for untuk setiap run length dalam encoded.
* r,p = rl[0], rl[1] : Deklarasi variabel r dan p dengan masing-masing nilainya diambil dari indeks 0 dan 1 dari rl.
* decoded.extend([p]\*r) : Menambahkan r elemen dengan nilai p ke dalam list decoded.
* dimg = np.array(decoded).reshape(shape) : Mengubah list decoded menjadi numpy array, kemudian direshape kembali ke bentuk asli gambar dengan menggunakan shape.
* return dimg : Mengembalikan hasil decoding dalam bentuk numpy array.
* dimg = RLE\_decode(encoded, shape) : Melakukan decoding pada gambar yang telah di-RLE encoding dengan menggunakan encoded dan shape yang diperoleh dari hasil encoding.
* show(dimg) : Menampilkan gambar hasil decoding menggunakan fungsi show().
  1. **Hitung Size Kompresi**



* Fungsi cv2.imwrite() digunakan untuk menyimpan hasil encoding dalam format file gambar TIFF, PNG, dan BMP



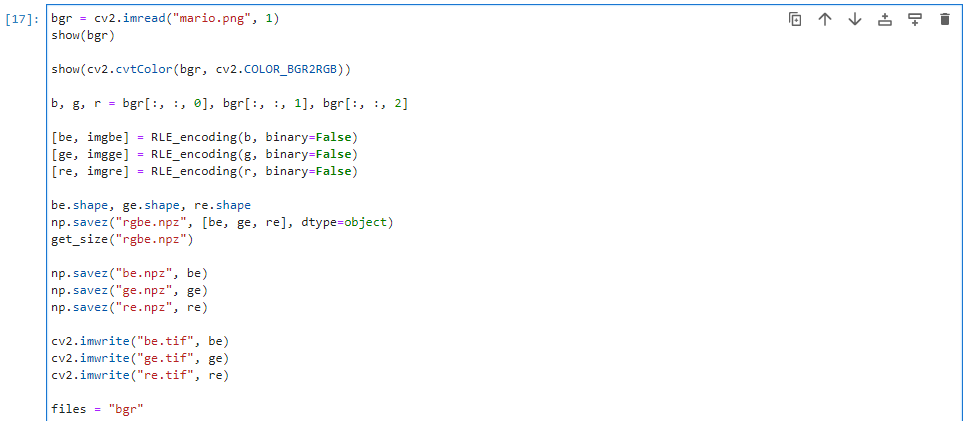
* Bagian ini digunakan untuk mengukur ukuran file dari beberapa file gambar yang telah dienkripsi menggunakan metode Run Length Encoding (RLE).
* Baris pertama membuat list files yang berisi nama-nama file gambar yang akan diuji ukurannya.
* Baris kedua menggunakan loop for untuk mengiterasi setiap file dalam list files.
* Baris ketiga mengambil ukuran file menggunakan fungsi get\_size(f), yang mengembalikan ukuran file dalam bytes.
* Baris keempat mencetak nama file dan ukurannya dalam format yang terstruktur menggunakan f-string.
* Dengan cara ini, kita dapat membandingkan ukuran file gambar sebelum dan setelah dienkripsi dengan metode RLE untuk mengetahui apakah teknik tersebut berhasil mengurangi ukuran file gambar atau tidak.

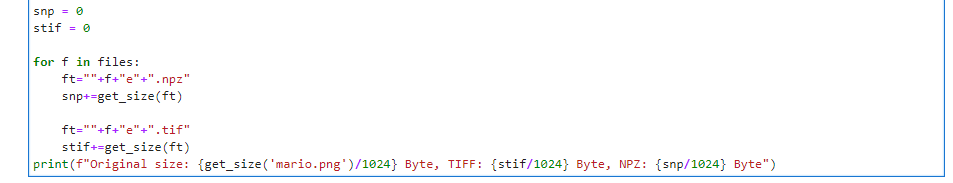




* Menampilkan gambar hasil decode RLE dan gambar asli

1. **RLE (*Run Length Encoding*) Gambar RGB**





bgr = cv2.imread("mario.png", 1)

show(bgr)

* Membaca gambar "mario.png" dalam mode BGR menggunakan OpenCV dan menampilkan gambar tersebut menggunakan fungsi show.

show(cv2.cvtColor(bgr, cv2.COLOR\_BGR2RGB))

* Mengonversi gambar BGR menjadi gambar RGB dan menampilkannya menggunakan fungsi show.

b, g, r = bgr[:, :, 0], bgr[:, :, 1], bgr[:, :, 2]

[be, imgbe] = RLE\_encoding(b, binary=False)

[ge, imgge] = RLE\_encoding(g, binary=False)

[re, imgre] = RLE\_encoding(r, binary=False)

* Memisahkan nilai komponen warna B, G, dan R dari gambar BGR menggunakan slicing dan mengekstraksi data RLE (Run-Length Encoding) dari setiap komponen warna tersebut menggunakan fungsi RLE\_encoding. Hasil dari fungsi RLE\_encoding disimpan dalam variabel be, imgbe, ge, imgge, re, dan imgre.

np.savez("rgbe.npz", [be, ge, re], dtype=object)

get\_size("rgbe.npz")

* Menyimpan data RLE yang sudah dihasilkan dari setiap komponen warna B, G, dan R ke dalam file "rgbe.npz" menggunakan fungsi np.savez. Selain itu, ukuran file "rgbe.npz" juga dihitung menggunakan fungsi get\_size.

np.savez("be.npz", be)

np.savez("ge.npz", ge)

np.savez("re.npz", re)

cv2.imwrite("be.tif", be)

cv2.imwrite("ge.tif", ge)

cv2.imwrite("re.tif", re)

* Menyimpan data RLE untuk setiap komponen warna B, G, dan R ke dalam file terpisah "be.npz", "ge.npz", dan "re.npz" menggunakan fungsi np.savez. Selain itu, data RLE juga disimpan dalam format gambar TIFF dengan nama file "be.tif", "ge.tif", dan "re.tif" menggunakan fungsi cv2.imwrite.

files = "bgr"

snp = 0

stif = 0

for f in files:

ft=""+f+"e"+".npz"

snp+=get\_size(ft)

ft=""+f+"e"+".tif"

stif+=get\_size(ft)

print(f"Original size: {get\_size('mario.png')/1024} Byte, TIFF: {stif/1024} Byte, NPZ: {snp/1024} Byte")

* Menghitung ukuran file "bge.npz" dan "bge.tif" menggunakan fungsi get\_size dan menampilkan ukuran file asli "mario.png", "bge.npz", dan "bge.tif".