Attempt at making a lossless compression algorithm for images

kips-sevenn

December 19, 2025

Context

File sharing today would not be possible without servers. Faced with the multitude of information being shared, several compression algorithms have been created. Two types can be distinguished:

- **Lossless data algorithms**: These allow data compression without altering the compressed data, for example:

Differential encoding

Differential encoding is a lossless compression method that converts a set of numbers into a sequence of differences relative to the previous one, except for the first element which is taken as the starting point: `delta([120,121,115,123,100])=[120,1,-6,8,-23]`

Context

File sharing today would not be possible without servers. Faced with the multitude of information being shared, several compression algorithms have been created. Two types can be distinguished:

- **Lossy data algorithms**: They compress data by sacrificing some details. These algorithms are generally used for visual and audio media: `.jpg` and `.mp3` formats are the most common.

Problem Statement

Problem Statement

How to efficiently reduce the space occupied by an image?

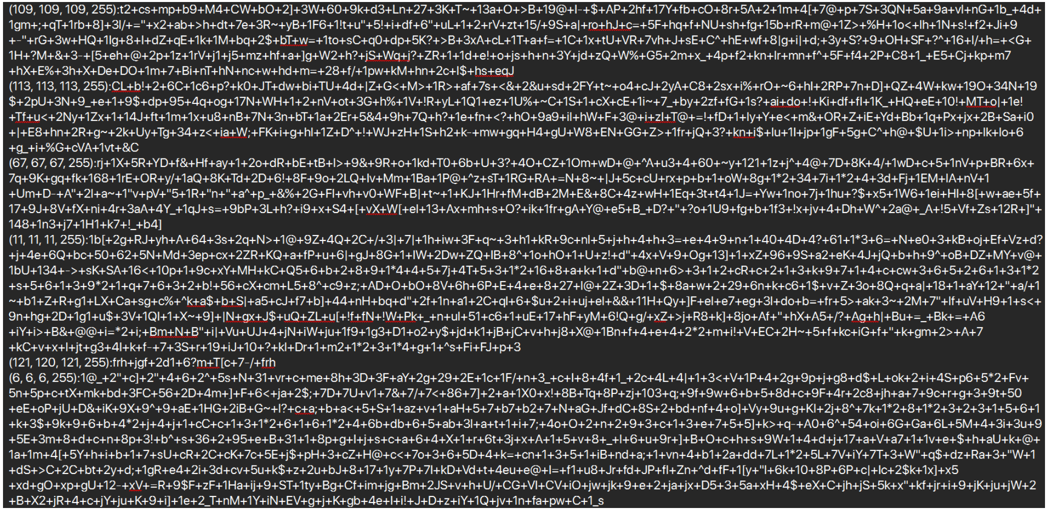
Compression Algorithm

This algorithm creates a dictionary that associates every color with a list of pixels possessing that color. Then, this dictionary is reduced in three steps:

1. Conversion of pairs into natural integers using Szudzik's elegant (bijective) function.
2. Application of differential encoding.
3. Conversion of the obtained list into a "sentence" using integers.
4. Change of numbering base: Switching to Base 77.

After these steps, the dictionary is saved in a `.txt` file which is then compressed into a `.xz` archive.

Compression Algorithm



```
(109, 109, 109, 255):t2+cs+mp+b9+M4+CW+bO+2]+3W+60+9k+d3+Ln+27+3K+T~+13a+O+>B+19@+l-+$+AP+2hf+17Y+fb+cO+8r+5A+2+1m+4[+7@+p+7S+3QN+5a+9a+vl+nG+1b_+4d+
1gm+;+qT+1rb+8]+3l/+="+x2+ab+>h+dt+7e+3R~+yB+1F6+1!t+u"+5!+i+df+6"+uL+1+2+rV+zt+15/+9S+a|+ro+hJ+c=+5F+hq+f+NU+sh+fg+15b+rR+m@+1Z>+%H+1o<+Ih+1N+s!+f2+Ji+9
+-"+rG+3w+HQ+1lg+8+l+dZ+qE+1k+1M+bq+2$+bT+w=+1to+sC+q0+dp+5K?+>B+3xA+cL+1T+a+f=+1C+1x+tU+VR+7vh+J+sE+C^+hE+wf+8|g+i|+d;+3y+S?+9+OH+SF+?^+16+l/+h=+<G+
1H+?M+&+3-+[5+eh+@+2p+1z+1rV+j1+j5+mz+hf+a+]g+W2+h?+jS+Wq+j?+ZR+1+1d+e!+o+js+h+n+3Y+jd+zQ+W%+G5+2m+x_+4p+f2+kn+lr+mn+f^+5F+f4+2P+C8+1_+E5+Cj+kp+m7
+hX+E%+3h+X+De+DO+1m+7+Bi+nT+hN+nc+w+hd+m=+28+f/+1pw+kM+hn+2c+l$+hs+eqJ
(113, 113, 113, 255):CL+b!+2+6C+1c6+p?+k0+JT+dw+bi+TU+4d+|Z+G<+M>+1R>+af+7s+<&+2&u+sd+2FY+t~+o4+cJ+2yA+C8+2sx+i%+rO+~6+hl+2RP+7n+D]+QZ+4W+kw+19O+34N+19
$+2pU+3N+9_+e+1+9$+dp+95+4q+og+17N+WH+1+2+nV+ot+3G+h%+1V+!R+yL+1Q1+ez+1U%+~C+1S+1+cX+cE+1i~+7_+by+2zf+fG+1s?+ai+do+!+Ki+df+fl+1K_+HQ+eE+10!+MT+o|+1e!
+Tf+u<+2Ny+1Zx+1+14J+ft+1m+1x+u8+nB+7N+3n+bT+1a+2Er+5&4+9h+7Q+h?+1e+fn+<?+hO+9a9+il+hW+F+3@+i+zl+T@+=!+fD+1+ly+Y+e<+m&+OR+Z+iE+Yd+Bb+1q+Px+jx+2B+Sa+i0
+|+E8+hn+2R+g~+2k+Uy+Tg+34+z<+ia+W;+FK+i+g+hl+1Z+D^+!+WJ+zH+1S+h2+k-+mw+gq+H4+gU+W8+EN+GG+Z>+1fr+jQ+3?+kn+i$+lu+1l+jp+1gF+5g+C^+h@+$U+1i>+np+lk+lo+6
+g_+i+%G+cVA+1vt+&C
(67, 67, 67, 255):rj+1X+5R+YD+f&+Hf+ay+1+2o+dR+bE+tB+l>+9&+9R+o+1kd+T0+6b+U+3?+4O+CZ+1Om+wD+@+^A+u3+4+60+~y+121+1z+j^+4@+7D+8K+4/+1wD+c+5+1nV+p+BR+6x+
7q+9K+gq+fk+168+1rE+OR+y/+1aQ+8K+Td+2D+6!+8F+9o+2LQ+lv+Mm+1Ba+1P@+^z+sT+1RG+RA+=N+8~+|J+5c+cU+rx+p+b+1+oW+8g+1*2+34+7i+1*2+4+3d+Fj+1EM+lA+nV+1
+Um+D-+A"+2l+a~+1"v+pV+"5+1R+"n+"+a^+p_+&%+2G+Fl+vh+v0+WF+B|+t~+1+KJ+1Hr+fM+dB+2M+E&+8C+4z+wH+1Eq+3t+t4+1J=+Yw+1no+7j+1hu+?$+x5+1W6+1ei+Hl+8[+w+ae+5f+
17+9J+8V+fX+ni+4r+3aA+4Y_+1qJ+s=+9bP+3L+h?+i9+x+S4+[+vX+W[+el+13+Ax+mh+s+O?+ik+1fr+gA+Y@+e5+B_+D?+"+?o+1U9+fg+b+1f3+!x+jv+4+Dh+W^+2a@+_A+!5+Vf+Zs+12R+]"+
148+1n3+j7+1H1+k7+!_+b4]
(11, 11, 11, 255):1b[+2g+RJ+yh+A+64+3s+2q+N>+1@+9Z+4Q+2C+/+3|+7|+1h+iw+3F+q~+3+h1+kR+9c+nl+5+j+h+4+h+3=+e+4+9+n+1+40+4D+4?+61+1*3+6=+N+e0+3+kB+oj+Ef+Vz+d?
+j+4e+6Q+bc+50+62+5N+Md+3ep+cx+2ZR+KQ+a+fP+u+6|+gJ+8G+1+lW+2Dw+ZQ+lB+8^+1o+hO+1+U+z!+d"+4x+V+9+Og+13]+1+xZ+96+9S+a2+eK+4J+jQ+b+h+9^+oB+DZ+MY+v@+
1bU+134+->+sK+SA+16<+10p+1+9c+xY+MH+kC+Q5+6+b+2+8+9+1*4+4+5+7j+4T+5+3+1*2+16+8+a+k+1+d"+b@+n+6>+3+1+2+cR+c+2+1+3+k+9+7+1+4+c+cw+3+6+5+2+6+1+3+1*2
+s+5+6+1+3+9*2+1+q+7+6+3+2+b!+56+cX+cm+L5+8^+c9+z;+AD+O+bO+8V+6h+6P+E+4+e+8+27+l@+2Z+3D+1+$+8a+w+2+29+6n+k+c6+1$+v+Z+3o+8Q+q+a|+18+1+aY+12+"+a/+1
~+b1+Z+R+g1+LX+Ca+sg+c%+^k+a$+b+S|+a5+cJ+f7+b]+44+nH+bq+d"+2f+1n+a1+2C+ql+6+$u+2+i+uj+el+&&+11H+Qy+]F+el+e7+eg+3l+do+b=+fr+5>+ak+3~+2M+7"+lf+uV+H9+1+s<+
9n+hg+2D+1g1+u$+3V+1Ql+1+X~+9]+|N+gx+J$+uQ+ZL+u[+!f+fN+!W+Pk+_+n+ul+51+c6+1+uE+17+hF+yM+6!Q+g/+xZ+>j+R8+k]+8jo+Af+"+hX+A5+/?+Ag+h|+Bu+=_+Bk+=+A6
+iY+i>+B&+@@+i=*2+i;+Bm+N+B"+i|+Vu+UJ+4+jN+iW+ju+1f9+1g3+D1+o2+y$+jd+k1+jB+jC+v+h+j8+X@+1Bn+f+4+e+4+2*2+m+i!+V+EC+2H~+5+f+kc+iG+f+"+k+gm+2>+A+7
+kC+v+x+l+jt+g3+4l+k+f-+7+3S+r+19+iJ+10+?+kl+Dr+1+m2+1*2+3+1*4+g+1+^s+Fi+FJ+p+3
(121, 120, 121, 255):frh+jgf+2d1+6?m+T[c+7-/+frh
(6, 6, 6, 255):1@_+2"+c]+2"+4+6+2^+5s+N+31+vr+c+me+8h+3D+3F+aY+2g+29+2E+1c+1F/+n+3_+c+l+8+4f+1_+2c+4L+4|+1+3<+V+1P+4+2g+9p+j+g8+d$+L+ok+2+i+4S+p6+5*2+Fv+
5n+5p+c+tX+mk+bd+3FC+56+2D+4m+]+F+6<+ja+2$;+7D+7U+v1+7&+7/+7<+86+7]+2+a+1X0+x!+8B+Tq+8P+zj+103+q;+9f+9w+6+b+5+8d+c+9F+4r+2c8+jh+a+7+9c+r+g+3+9t+50
+eE+oP+jU+D&+iK+9X+9^+9+aE+1HG+2iB+G~+l?+c+a;+b+a<+5+S+1+az+v+1+aH+5+7+b7+b2+7+N+aG+Jf+dC+8S+2+bd+nf+4+o]+Vy+9u+g+Kl+2j+8^+7k+1*2+8+1*2+3+2+3+1+5+6+1
+k+3$+9k+9+6+b+4*2+j+4+j+1+cC+c+1+3+1*2+6+1+6+1*2+4+6b+db+6+5+ab+3l+a+t+1+i+7;+4o+O+2+n+2+9+3+c+1+3+e+7+5+5]+k>+q-+A0+6^+54+oi+6G+Ga+6L+5M+4+3i+3u+9
+5E+3m+8+d+c+n+8p+3!+b^+s+36+2+95+e+B+31+1+8p+g+l+j+s+c+a+6+4+X+1+r+6t+3j+x+A+1+5+v+8+_+l+6+u+9r+]+B+O+c+h+s+9W+1+4+d+j+17+a+V+a7+1+1v+e+$+h+aU+k+@+
1a+1m+4[+5Y+h+i+b+1+7+sU+cR+2C+cK+7c+5E+j$+pH+3+cZ+H@+c<+7o+3+6+5D+4+k=+cn+1+3+5+1+iB+nd+a;+1+vn+4+b1+2a+dd+7L+1*2+5L+7V+iY+7T+3+W"+q$+dz+Ra+3+"W+1
+dS+>C+2C+bt+2y+d;+1gR+e4+2i+3d+cv+5u+k$+z+2u+bJ+8+17+1y+7P+7l+kD+Vd+t+4eu+e@+l=+f1+u8+Jr+fd+JP+fl+Zn+^d+fF+1[y+"l+6k+10+8P+6P+c|+lc+2$k+1x]+x5
+xd+gO+xp+gU+12-+xV+=R+9$F+zF+1Ha+ij+9+ST+1ty+Bg+Cf+im+jg+Bm+2JS+v+h+U/+CG+Vl+CV+iO+jw+jk+9+e+2+ja+jx+D5+3+5a+xH+4$+eX+C+jh+jS+5k+x"+kf+jr+i+9+jK+ju+jW+2
+B+X2+jR+4+c+jY+ju+K+9+i]+1e+2_T+nM+1Y+iN+EV+g+j+K+gb+4e+l+i!+J+D+z+iY+1Q+jv+1n+fa+pw+C+1_s
```

Figure: Extract from the .txt file (image_test_8)

Decompression Algorithm

This algorithm follows this specific order:

1. Decompression of the `.xz` archive and recovery of the dictionary.
2. Return to Base 10.
3. Decoding of the sentence.
4. Reciprocal of Szudzik's function.
5. Reconstruction of the image.

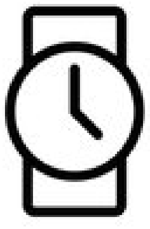
Results for: image_test_1.jpg

Dimensions: 217x232
Time elapsed for compression: 00h:00m:02s
Time elapsed for decompression: 00h:00m:00s
Failure | Original size/Compressed size: 6049/26224

Results for: image_test_2.png

Dimensions: 275x183
Time elapsed for compression: 00h:00m:06s
Failure | Original size/Compressed size: 5228/9276
Time elapsed for decompression: 00h:00m:00s

Results for: image_test_3.jpg

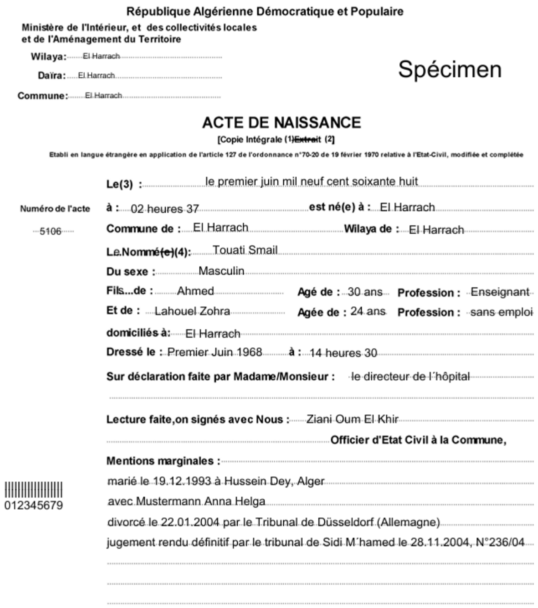


Dimensions: 99x137
Time elapsed for compression: 00h:00m:00s
Failure | Original size/Compressed size: 6184/6616
Time elapsed for decompression: 00h:00m:00s

Results for: image_test_4.jpg

Dimensions: 225x225
Time elapsed for compression: 00h:00m:04s
Failure | Original size/Compressed size: 10917/35604
Time elapsed for decompression: 00h:00m:00s

Results for: image_test_6.png



République Algérienne Démocratique et Populaire

Ministère de l'Intérieur, et des collectivités locales
et de l'Aménagement du Territoire

Wilaya: El Harrach
Daïra: El Harrach
Commune: El Harrach

Spécimen

ACTE DE NAISSANCE

[Copie Intégrale (1) ~~Extrait~~ (2]

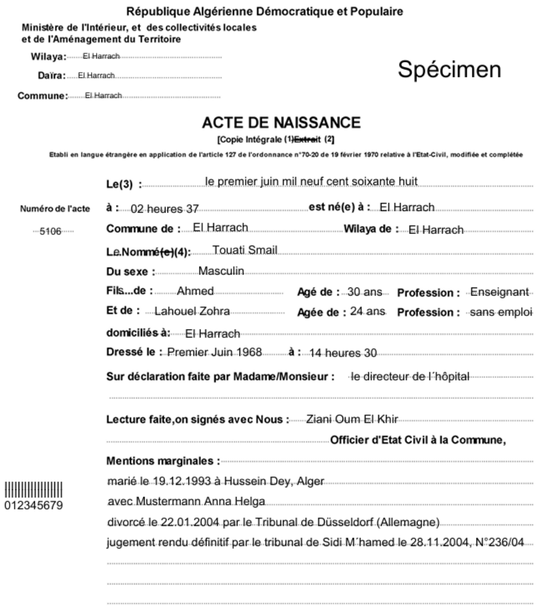
Etabli en langue étrangère en application de l'article 127 de l'ordonnance n°70-20 de 19 février 1970 relative à l'Etat-Civil, modifiée et complétée

Numéro de l'acte 5106

Le(3) : le premier juin mil neuf cent soixante huit
à : 02 heures 37 **est né(e) à :** El Harrach
Commune de : El Harrach **Wilaya de :** El Harrach
Le Nommé~~(e)~~(4): Touati Smail
Du sexe : Masculin
Fils...de : Ahmed **Agé de :** 30 ans **Profession :** Enseignant
Et de : Lahouel Zohra **Agée de :** 24 ans **Profession :** sans emploi
domiciliés à: El Harrach
Dressé le : Premier Juin 1968 **à :** 14 heures 30
Sur déclaration faite par Madame/Monsieur : le directeur de l´hôpital

Lecture faite,on signés avec Nous : Ziani Oum El Khir
Officier d'Etat Civil à la Commune,
Mentions marginales :
marié le 19.12.1993 à Hussein Dey, Alger
avec Mustermann Anna Helga
divorcé le 22.01.2004 par le Tribunal de Düsseldorf (Allemagne)
jugement rendu définitif par le tribunal de Sidi M´hamed le 28.11.2004, N°236/04

012345679



République Algérienne Démocratique et Populaire

Ministère de l'Intérieur, et des collectivités locales
et de l'Aménagement du Territoire

Wilaya: El Harrach
Daïra: El Harrach
Commune: El Harrach

Spécimen

ACTE DE NAISSANCE

[Copie Intégrale (1) ~~Extrait~~ (2]

Etabli en langue étrangère en application de l'article 127 de l'ordonnance n°70-20 de 19 février 1970 relative à l'Etat-Civil, modifiée et complétée

Numéro de l'acte 5106

Le(3) : le premier juin mil neuf cent soixante huit
à : 02 heures 37 **est né(e) à :** El Harrach
Commune de : El Harrach **Wilaya de :** El Harrach
Le Nommé~~(e)~~(4): Touati Smail
Du sexe : Masculin
Fils...de : Ahmed **Agé de :** 30 ans **Profession :** Enseignant
Et de : Lahouel Zohra **Agée de :** 24 ans **Profession :** sans emploi
domiciliés à: El Harrach
Dressé le : Premier Juin 1968 **à :** 14 heures 30
Sur déclaration faite par Madame/Monsieur : le directeur de l´hôpital

Lecture faite,on signés avec Nous : Ziani Oum El Khir
Officier d'Etat Civil à la Commune,
Mentions marginales :
marié le 19.12.1993 à Hussein Dey, Alger
avec Mustermann Anna Helga
divorcé le 22.01.2004 par le Tribunal de Düsseldorf (Allemagne)
jugement rendu définitif par le tribunal de Sidi M´hamed le 28.11.2004, N°236/04

012345679

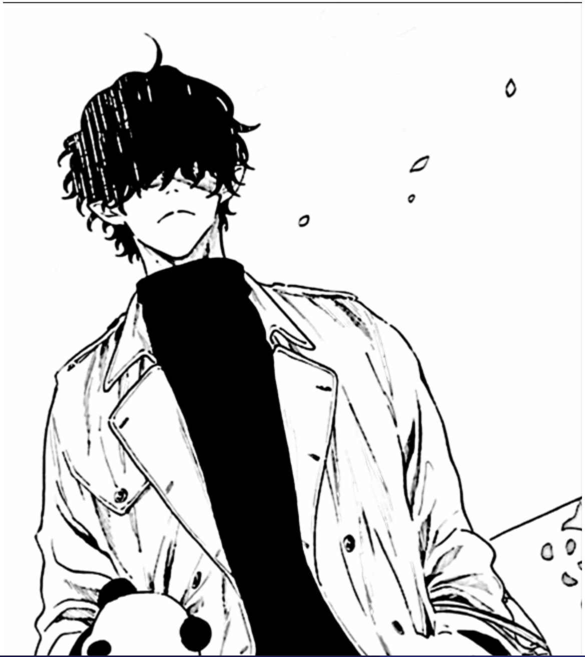
Results for: image_test_6.png

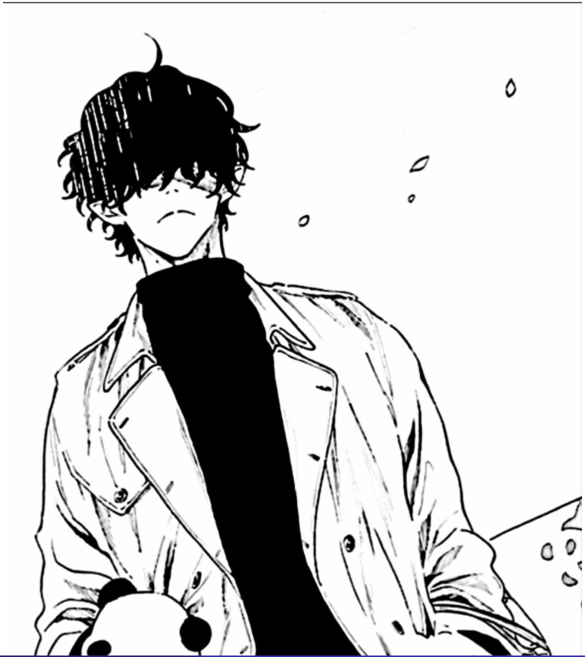
Dimensions: 691x978
Time elapsed for compression: 00h:15m:13s
Success | Original size/Compressed size: 214216/127788
Time elapsed for decompression: 00h:00m:01s

Results for: image_test_9.jpg

Dimensions: 710x948
Time elapsed for compression: 00h:01m:22s
Failure | Original size/Compressed size: 83948/445940
Time elapsed for decompression: 00h:00m:01s

Results for: image_test_9_prime.png





Results for: image_test_9_prime.png

Dimensions: 579x782
Time elapsed for compression: 00h:05m:23s
Success | Original size/Compressed size: 236863/129964
Time elapsed for decompression: 00h:00m:00s

Results Analysis

It seems that overly small or large dimensions affect information redundancy. For an optimal algorithm, the "perfect resolution" must be found. Based on the successes, we can estimate the target pixel count to be in the neighborhood of the interval [499000, 676000], i.e., resolutions around 700 x 900. Also, the presence of colors is unnecessary as it does not prevent character recognition but consumes space.

Unimplemented Ideas

Due to time constraints, the following ideas could not be added to the algorithm:

Idea: Line Reduction

If an entire line (or segment, to be correct) has the same color, it can be reduced by replacing the elements that are not at the extremities with a symbol indicating a fill.
Assuming an image of dimensions $7 \times$ `height` and considering the following extract:
`{(255,255,255,255):[(0,0),(0,1),(0,2),(0,3),(0,4),(0,5),(0,6)]}` could become:
`{(255,255,255,255):['(0,0)-(0,6)']}`
More generally, we could delimit color zones and only record boundary pixels.

Unimplemented Ideas

Due to time constraints, the following ideas could not be added to the algorithm:

Idea: Anti-aliasing Removal

When moving from one color zone to another, we observe pixels whose color is a mixture of the two zones' colors. This is done to improve visual quality (anti-aliasing).
We could eliminate these "transition pixels" for compression and then restore them using an anti-aliasing filter during decompression, resulting in less information to record.

Unimplemented Ideas

Due to time constraints, the following ideas could not be added to the algorithm:

Idea: Anti-aliasing Removal

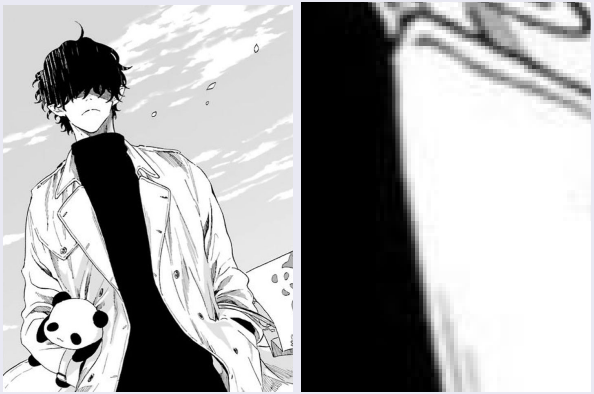


Figure: Illustration of anti-aliasing

Project Status

Discontinuation

Ultimately, I abandoned this project due to difficulties in implementation and limited success regarding the compression ratios achieved. Note that I tried optimizing the code, so the compression times might differ from the results shown.

Open Source Contribution

I created this GitHub repository to document the attempt and the methodology used. The goal is to allow anyone interested to:

- Analyze the current code and approach.
- Understand the limitations encountered.
- Potentially continue the project or use it as a base for improvement.

Note: This was my very first coding project, and despite the challenges, I had a lot of fun building it!

Python Code: Compression Algorithm I

```python
from PIL import Image
from BaseN import DecToBaseN #github/harrisonlingren
from re import findall
import lzma

image_name="image_name"
im = Image.open(image_name)
```

Python Code: Compression Algorithm II

```
def reindexation(pixel: tuple) -> int:
    if int(pixel[0])<int(pixel[1]):
        return int(pixel[1])**2+int(pixel[0])
    else:
        return int(pixel[0])**2+int(pixel[0])+int(pixel[1])
```

A bijection between $\mathbb{N}^2$ and $\mathbb{N}$, we choose Szudzik's elegant pairing function:

$$f(x,y) = \begin{cases} y^2 + x \text{ if } x < y \\ x^2 + x + y \text{ if } x \geq y \end{cases}$$

Python Code: Compression Algorithm III

```python
def delta(positions: list) -> list:
    for i in range(len(positions)):
      positions[i]=reindexation(positions[i])
    positions.sort()

    for i in range(1,len(positions)):
        positions[-i]-=positions[-(i+1)]
    for i in range(1,len(positions)):
        positions[i]="+" + str(positions[i])
    return positions
```

This is an implementation of differential encoding.

Python Code: Compression Algorithm IV

```python
def reecriture(positions:list)->str:
    """
    The goal is to create a string that takes up less space in the txt file
    and is easy to decode. Algorithm illustration:
    [100,105,108,111,120,155,190,225] --delta--> [100,'+5','+3','+3','+12','+35','+35','+35']
    --rewrite--> '100+5+3*2+12+35*3'

    In this optimized version, we use the two-pointer method to reduce complexity from O(n^2) to
         O(n).
    """
    left, right = 0, 1
    phrase_fractionnee = []

    while right < len(positions):
        if positions[right] == positions[left]:
            right += 1
        else:
            if right - left == 1:
                phrase_fractionnee.append(str(positions[left]))
            else:
                phrase_fractionnee.append(f'{positions[left]}*{right-left}')
            left = right
            right += 1
    # Process the last block
    if right - left == 1:
        phrase_fractionnee.append(str(positions[left]))
    else:
        phrase_fractionnee.append(f'{positions[left]}*{right-left}')
    return ''.join(phrase_fractionnee)
```

Python Code: Compression Algorithm IV

The goal is to make a sentence that will take up less space in the .txt file and be easy to decode, illustration: `[100,105,108,111,120,155,190,225]`–delta–>`[100,'+5','+3','+3','+12','+35','+35','+35']`–rewrite–>`'100+5+3*2+12+35*3'`

Python Code: Compression Algorithm V

```python
#Dictionary Creation {color:color_list}
(width, height) = im.size
color_positions = {}
for x in range(width):
    for y in range(height):
        pixel_color = im.getpixel((x, y))
        if pixel_color not in color_positions:
            color_positions[pixel_color]=[]
            color_positions[pixel_color].append((x, y))
        else: color_positions[pixel_color].append((x, y))
```

Python Code: Compression Algorithm VI

```python
#Dictionary Reduction:
for color in color_positions.keys():
    color_positions[color]=reecriture(delta(color_positions[color]))
digitset='0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"$%&-/;<>=?@[]^_|~'
for color in color_positions.keys():
    temp_list=findall(r"\d+|[*+]",color_positions[color])
    for i in range(len(temp_list)):
        if temp_list[i].isdigit(): temp_list[i]=DecToBaseN(int(temp_list[i]), digitset)
    color_positions[color]=''.join(temp_list)
```

Python Code: Compression Algorithm VII

```
#Writing information to the .txt file
#We add dimensions for reconstruction
with open('information.txt', 'w', encoding="ascii") as fichier:
    for color, positions in color_positions.items(): fichier.write(f"{color}:{positions}\n")
    fichier.write(f"width,height:{width},{height}")

#Compression to .xz
with open("information.txt", "rb") as information, lzma.open("information.txt.xz", "wb", preset
    =9) as compressé:
    compressé.write(information.read())
```

Python Code: Decompression Algorithm I

```python
from PIL import Image
from BaseN import BaseNToDec
from math import sqrt,floor
import lzma
```

Python Code: Decompression Algorithm II

```python
def recip_reecriture(entier: int) -> tuple:
    s = floor(sqrt(entier))
    if entier - s*s < s: (x, y) = (entier - s*s, s)
    else: (x, y) = (s, entier - s*s - s)
    return (x, y)
```

This is the reciprocal of the elegant function:

$$\begin{cases} f(z) = (z - \lfloor\sqrt{z}\rfloor^2, \lfloor\sqrt{z}\rfloor) \text{ if } z - \lfloor\sqrt{z}\rfloor^2 < \\ f(z) = (\lfloor\sqrt{z}\rfloor, z - \lfloor\sqrt{z}\rfloor^2 - \lfloor\sqrt{z}\rfloor) \text{ else} \end{cases}$$

Python Code: Decompression Algorithm III

```
#Reading the archive and creating the dictionary
with lzma.open("information.txt.xz", "rb") as archive, open("decompressé.txt", "wb") as
     fichier_txt: fichier_txt.write(archive.read())
color_positions={}
with open('decompressé.txt', 'r', encoding="ascii") as fichier:
    for ligne in fichier:
        resultat = ligne.split(':')
        resultat[1]=resultat[1].replace('\n','')
        color_positions[resultat[0]]=resultat[1]
(width,height)=(int(color_positions['width,height'].split(",")[0]),int(color_positions['width,
     height'].split(",")[1]))
del color_positions['width,height']
```

Python Code: Decompression Algorithm IV

```python
    #Sentence in base 10:
digitset='0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"$%&-/;<>=?@[]^_|~'
for color in color_positions.keys():
    result = ''
    current_number = ''

    for char in color_positions[color]:
        if char in ('*', '+'):
            if current_number:
                result += BaseNToDec(current_number, digitset)
                current_number = ''
            result += char
        else:
            current_number += char

    if current_number:  # For the last number
        result += BaseNToDec(current_number, digitset)

    color_positions[color] = result
```

Python Code: Decompression Algorithm V

```
for color in color_positions.keys():
    # String splitting
    nombres = [n for n in color_positions[color].split('+') if n]
    decompresse = []

    #Reconstruction of the integer list without '*'
    for num in nombres:
        if '*' in num:
            valeur, repetitions = map(int, num.split('*'))
            decompresse.extend([valeur] * repetitions)
        else:
            decompresse.append(int(num))
    # Delta decoding
    for i in range(1, len(decompresse)):
        decompresse[i] += decompresse[i-1]
    color_positions[color] = decompresse
```

Python Code: Decompression Algorithm VI

```python
#Return to N^2:
for color in color_positions.keys():
    coordonnees_pixels = []
    for i in range(len(color_positions[color])):
        valeur = int(color_positions[color][i])
        coordonnees_pixels.append(recip_reecriture(valeur))
    color_positions[color] = coordonnees_pixels
```

Python Code: Decompression Algorithm VII

```python
#Image Reconstruction
img = Image.new("RGBA", (width, height), color=(0,0,0,0))
color_positions = {eval(k): v for k, v in color_positions.items()}
for color in color_positions.keys():
    for i in range(len(color_positions[color])):
        coord = color_positions[color][i]
        img.putpixel(coord, color)
img.save("final_image.png")
```

Python Code: DecToBaseN

```python
# x = base 10 [str], d = set of ordered unique 'digits' [str]
def DecToBaseN(x, d):
    # k = helper, b = result, n = size of set
    k = int(x)
    b = ''
    n = len(d)

    # check if x is within n
    if k < n - 1: return d[k]
    # if not, divide and mod by n until x < n
    else:
        while k > n - 1:
            r = k % n #remainder
            b = d[r] + b #done for the order of the base
            k //= n
        if k > 0: #for the case where the divider isn't null
            #b += d[k]
            b = d[k] + b
    return b
```

Python Code: BaseNToDec

```
# x = base N number as [str], d = set of ordered unique 'digits' [str]
def BaseNToDec(x, d):
    # b = result, n = num digits in x, m = size of set
    b = 0
    n = len(x)
    m = len(d)

    # for sym in x, get base 10 index of sym and multiply by m^i ( 0 < i < n )
    j = 0
    for i in range(n - 1, -1, -1):
        a = int(d.index(x[i]))
        b += ( a * ( m ** j ) )
        j += 1
        #classical mathematical formula to comeback to base 10
    return str(b)
```