ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

BÁO CÁO ĐỒ ÁN

Tính toán đa phương tiện – CS232.N21

IMAGE COMPRESSION

GIẢNG VIÊN HƯỚNG DẪN:

ThS. Đỗ Văn Tiến

SINH VIÊN THỰC HIỆN:

Nguyễn Huỳnh Minh Triết – 21520497

Nguyễn Minh Thư – 21520472

Lê Châu Giang – 21520213

Ngô Phúc Danh – 21521924

Tp. Hồ Chí Minh, 07/2023

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn sâu sắc nhất đến **thầy Đỗ Văn Tiến** là giảng viên môn **Tính toán đa phương tiện** đã hết lòng hướng dẫn, giúp đỡ đồng thời đưa ra những góp ý chỉ dạy cho chúng em những kiến thức còn thiếu và tạo mọi điều kiện để chúng em những kiến thức còn thiếu và tạo mọi điều kiện để chúng em hoàn thành đồ án cuối kỳ một cách tốt nhất.

Trong thời gian qua, tham gia lớp học **Tính toán đa phương tiện** của thầy, chúng em đã có thêm cho mình nhiều kiến thức quý báu từ kỹ thuật crawl dữ liệu, xử lý dữ liệu XML cho đến các kỹ thuật nén dữ liệu,.. đây là hành trang quý báu cho chúng em có thể vững bước không chỉ ở môi trường đại học mà còn là sự chuẩn bị cần thiết cho việc đi làm về sau.

Bộ môn **Tính toán đa phương tiện** là môn học quan trọng, vô cùng bổ ích và có tính ứng dụng cao đối với sinh viên UIT nói chung và sinh viên Khoa học máy tính nói riêng. Tuy nhiên trong quá trình thực hiện báo cáo đồ án **Image compression**, do kiến thức chuyên môn còn hạn chế nên chúng em vẫn còn nhiều thiếu sót khi tìm hiểu và thực hiện đồ án. Nhóm chúng em rất mong nhận được sự quan tâm, góp ý của thầy về đồ án của nhóm để chúng em có thể ngày càng hoàn thiện hơn nữa.

Chúng em xin chân thành cảm ơn thầy!

CHƯƠNG I. THUẬT TOÁN NÉN ẢNH JPEG

PHẦN 1. GIỚI THIỆU

JPEG (Joint Photographic Experts Group) là một chuẩn nén hình ảnh kỹ thuật số. JPEG được sử dụng rộng rãi để nén và lưu trữ hình ảnh số trên máy tính và trên internet.

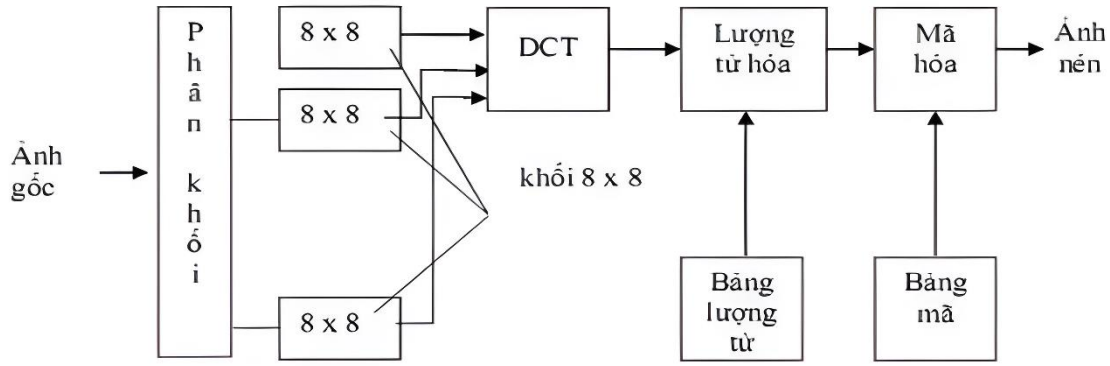
Một tệp tin hình ảnh JPEG có phần mở rộng là ".jpg" hoặc ".jpeg" và có thể mở và xem trên hầu hết các trình duyệt web, chương trình xem ảnh, và các ứng dụng hỗ trợ hình ảnh.

Phương pháp nén ảnh JPEG là thuật toán nén hình ảnh mất mát (lossy compression), thuật toán này làm giảm kích thước tệp tin hình ảnh nhưng vẫn giữ được mức chất lượng hình ảnh chấp nhận được. Phương pháp nén này loại bỏ những thông tin không cần thiết và giảm chi tiết hình ảnh nhằm giảm dung lượng tệp tin.

Do đó, JPEG thường được sử dụng cho hình ảnh có màu sắc phức tạp và chi tiết, chẳng hạn như hình ảnh kỹ thuật số, ảnh chụp từ máy ảnh, và hình ảnh trên trang web.

PHẦN 2. THUẬT TOÁN

2.1. TỔNG QUAN QUY TRÌNH NÉN



Các bước nén ảnh bao gồm:

- Chuyển đổi không gian màu
- Trung tâm dữ liệu
- Chia khối và biến đổi Cosine rời rạc (DCT)
- Lượng tử hóa
- Mã hóa Entropy
- Lưu trữ và kết quả

2.2. CHI TIẾT QUY TRÌNH NÉN

2.2.1. CHUYỂN ĐỔI KHÔNG GIAN MÀU

Phương pháp nén JPEG thường chuyển không gian màu RGB sang không gian màu YUV trước khi áp dụng quá trình nén. Có một số lý do chính cho việc chuyển đổi này:

Tính tách biệt của thông tin màu sắc và thông tin độ sáng: Trong không gian màu RGB, mỗi điểm ảnh được biểu diễn bởi ba thành phần màu sắc: đỏ (Red), xanh lá cây (Green) và xanh dương (Blue). Tuy nhiên, trong một ảnh, thông tin

về độ sáng (luminance) thường quan trọng hơn so với thông tin về màu sắc. Bằng cách chuyển đổi sang không gian màu YUV, ta có thể tách biệt thông tin độ sáng (Y) và thông tin màu sắc (U, V), và ưu tiên nén thông tin một cách hiệu quả hơn.

Độ nhạy của mắt người đối với màu sắc và độ sáng: Mắt người thường nhạy hơn với thông tin độ sáng so với thông tin màu sắc. Bằng cách áp dụng lượng tử hóa và nén dữ liệu trong không gian màu YUV, ta có thể sử dụng các giá trị lượng tử khác nhau cho thành phần độ sáng (Y) và thành phần màu sắc (U, V), từ đó tối ưu hóa quá trình nén và giảm lượng dữ liệu cần lưu trữ.

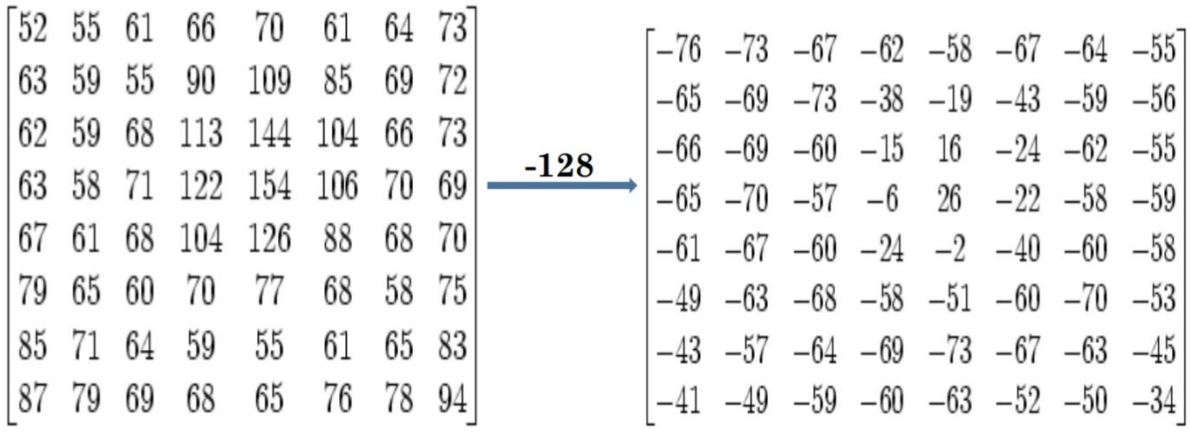
Giảm sự tương quan không cần thiết: Các thành phần U và V trong không gian màu YUV biểu thị mức độ khác biệt của màu sắc trong ảnh. Tuy nhiên, sự khác biệt này thường không cần thiết chi tiết so với thông tin độ sáng chính (Y). Bằng cách áp dụng lượng tử hóa và nén trên các thành phần U và V, ta có thể loại bỏ các chi tiết không cần thiết và giảm kích thước tệp tin một cách hiệu quả.

Tóm lại, việc chuyển đổi từ không gian màu RGB sang không gian màu YUV trong quá trình nén JPEG cho phép tách biệt thông tin độ sáng và thông tin màu sắc, tận dụng độ nhạy khác nhau của mắt người và giảm sự tương quan không cần thiết. Điều này giúp cải thiện hiệu suất nén và đạt được kích thước tệp tin nhỏ hơn mà vẫn giữ được chất lượng hình ảnh chấp nhận được.

2.2.2. TRUNG TÂM DỮ LIỆU

Trước khi bước vào quá trình DCT, ta thường thực hiện trung tâm dữ liệu (data centering) hay còn gọi là trung tâm mức xám (gray level shifting). Quá trình này nghĩa là ta lấy ma trận điểm ảnh trừ cho 128.

Ví dụ:



Bước này có tác dụng như sau như sau:

Tối ưu hoá biểu diễn: Khi sử dụng biến đổi DCT, việc trừ đi 128 từ mỗi phần tử trong ma trận giúp tối ưu hoá biểu diễn thông tin hình ảnh. Bằng cách trừ đi giá trị trung tâm 128, các giá trị DCT sẽ tập trung xung quanh giá trị 0, đồng thời giảm mức độ biến thiên của các hệ số DCT. (Thông tin về phép biến đổi DCT sẽ được lý giải cụ thể sau)

Giảm ảnh hưởng của hệ số DC: Hệ số DC, là thành phần ở góc trái trên cùng trong khối DCT, thể hiện mức độ thay đổi trung bình của khối 8x8. Bằng cách trừ 128 trước khi tính toán DCT, giá trị hệ số DC sau đó sẽ thể hiện sự khác biệt với mức xám trung bình của khối, giảm ảnh hưởng của giá trị DC đến hình ảnh chung. (Thông tin về hệ số DC sẽ được

Tương thích với việc biểu diễn âm và dương: Quá trình trừ 128 giúp chuyển đổi miền giá trị từ [0, 255] sang miền giá trị âm và dương, từ [-128, 127]. Điều này hữu ích cho việc biểu diễn thông tin âm và dương trong quá trình biến đổi DCT và các bước mã hóa tiếp theo.

2.2.3. CHIA KHỐI VÀ BIẾN ĐỔI COSINE RỜI RẠC (DCT)

2.2.3.1. GIỚI THIỆU VỀ DCT

Thuật toán DCT (Discrete Cosine Transform) là một thuật toán biến đổi tín hiệu số để chuyển đổi một tín hiệu trong miền thời gian sang miền tần số. Nó được sử dụng rộng rãi trong xử lý tín hiệu và nén hình ảnh và âm thanh.

Trong nén hình ảnh và âm thanh, DCT thường được sử dụng để giảm kích thước dữ liệu bằng cách chuyển đổi các khối dữ liệu tín hiệu thành dạng biểu diễn tần số. Sau đó, các hệ số tần số có thể được lưu trữ hoặc truyền đi một cách hiệu quả hơn so với dữ liệu ban đầu. Khi cần phục hồi tín hiệu gốc, DCT ngược được áp dụng để chuyển đổi lại từ miền tần số sang miền thời gian. DCT có nhiều biến thể, phổ biến nhất là DCT loại II (DCT-II), còn gọi là DCT chuẩn. Ngoài ra, còn có các biến thể như DCT loại I, III, IV, và DCT ngược (IDCT).

2.2.3.2. BIẾN ĐỔI DCT TRONG JPEG

Phương pháp nén ảnh JPEG sử dụng thuật toán DCT-II (Discrete Cosine Transform type II). Nó được sử dụng rộng rãi trong nén hình ảnh và âm thanh.
DCT-II thực hiện biến đổi một chuỗi dữ liệu thời gian diskret thành một chuỗi dữ liệu tần số diskret bằng cách sử dụng hàm cosine. Kết quả là một tập hệ số tần số biểu diễn các thành phần tần số khác nhau trong tín hiệu ban đầu.
Công thức biến đổi DCT-II cho một chuỗi dữ liệu đầu vào có độ dài N là:

$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(i).\Lambda(j).cos\left[\frac{\pi.u}{2.N}(2i+1)\right] cos\left[\frac{\pi.v}{2.M}(2j+1)\right].f(i,j)$$

Trong đó:

F(u,v): Giá trị DCT tại tần số (u,v)

N, M: 2 kích thước của ma trận đầu vào

Λ(i), Λ(j): Hệ số chuẩn hóa

f(i,j): Giá trị tại điểm (i,j) của ma trận đầu vào

Áp dụng công thức trên, ta được ma trận mới, gọi là ma trận DCT:

Ví dụ:

130	132	132	129	133	133	134	135
135	133	133	131	133	137	138	136
132	134	133	136	139	142	137	137
137	136	136	135	135	136	135	138
139	138	139	135	136	139	137	140
134	136	137	136	134	136	136	143
137	133	138	136	136	136	139	146
139	137	138	134	133	138	140	146

DCT →

1089.1	-10.4	8.1	-2.4	-0.4	-2.2	4.9	0
-12.5	0.6	-5.9	6.3	-2.9	-1.0	-2.8	0.4
-4.9	3.8	3.9	-1.2	-0.5	0.9	-0.3	1.8
-4.2	1.2	0.7	-3.5	0.3	-0.8	0	1.3
0.6	4.2	2.7	-0.4	-1.1	-0.8	0.4	0.8
2.6	-1.2	-2.4	-1.9	2.9	-0.6	-0.5	0.8
1.1	-1.7	-3.1	-0.4	-0.8	-2.5	-0.9	2.3
3.1	-1.7	-2.6	2.0	-0.1	-0.7	0.6	0.5

Việc sử dụng DCT-II trong nén ảnh giúp giảm kích thước dữ liệu bằng cách tận dụng tính chất thống kê của hình ảnh.

DCT-II chuyển đổi một khối hình ảnh từ miền thời gian sang miền tần số. Kết quả của DCT-II là một tập hệ số tần số, trong đó các hệ số tương ứng với các thành phần tần số khác nhau trong khối hình ảnh ban đầu.

Các hệ số trong ma trận DCT được sắp xếp theo mức độ quan trọng của chúng. Thông thường, các hệ số tần số cao có đóng góp ít hơn và thường có giá trị gần bằng 0, trong khi các hệ số tần số thấp có đóng góp nhiều hơn và thường có giá trị lớn hơn. Khi chúng ta giữ lại một số lượng hệ số tần số quan trọng nhất và loại bỏ các hệ số tần số không quan trọng, chúng ta có thể giảm kích thước dữ liệu một cách đáng kể.

2.2.3.3. LƯỢNG TỬ HÓA

Quá trình lượng tử hóa là quá trình chia các giá trị hệ số DCT (Discrete Cosine Transform) cho các giá trị lượng tử để giảm kích thước dữ liệu và tạo ra một mức độ mất mát thông tin xác định.

Cụ thể, quá trình lượng tử hóa trong JPEG bao gồm các bước sau:

- Xác định bảng lượng tử: Đầu tiên, được xác định một bảng lượng tử (quantization table) cho mỗi thành phần DCT. Bảng này bao gồm các giá trị lượng tử, thường được tạo ra trước quá trình nén và được điều chỉnh dựa trên mức độ chất lượng mong

muốn. Giá trị lượng tử cao sẽ dẫn đến mất mát thông tin cao hơn và kích thước tệp tin nhỏ hơn.

- Chia các hệ số DCT cho giá trị lượng tử: Các hệ số DCT thu được từ bước biến đổi DCT sẽ được chia cho các giá trị lượng tử tương ứng từ bảng lượng tử. Quá trình này giới hạn giá trị của hệ số DCT và làm giảm số lượng thông tin được lưu trữ.
- Làm tròn giá trị lượng tử: Sau khi chia cho giá trị lượng tử, các giá trị thu được có thể không phải là các số nguyên. Do đó, các giá trị này sẽ được làm tròn thành các số nguyên gần nhất. Quá trình làm tròn này cũng góp phần giảm kích thước dữ liệu.

Kết quả của quá trình lượng tử hóa là các giá trị hệ số lượng tử đã được làm tròn. Những giá trị này thể hiện mức độ đóng góp của các tần số khác nhau trong ảnh và thường được biểu diễn bằng các số nguyên có kích thước nhỏ hơn so với hệ số DCT gốc. Quá trình lượng tử hóa là một phần quan trọng trong quá trình nén JPEG, nó giúp giảm kích thước tệp tin hình ảnh mà vẫn duy trì được mức chất lượng chấp nhận được.

Công thức lượng tử hóa:

$$Fq(u,v)=[F(u,v)/Q(u,v)]$$

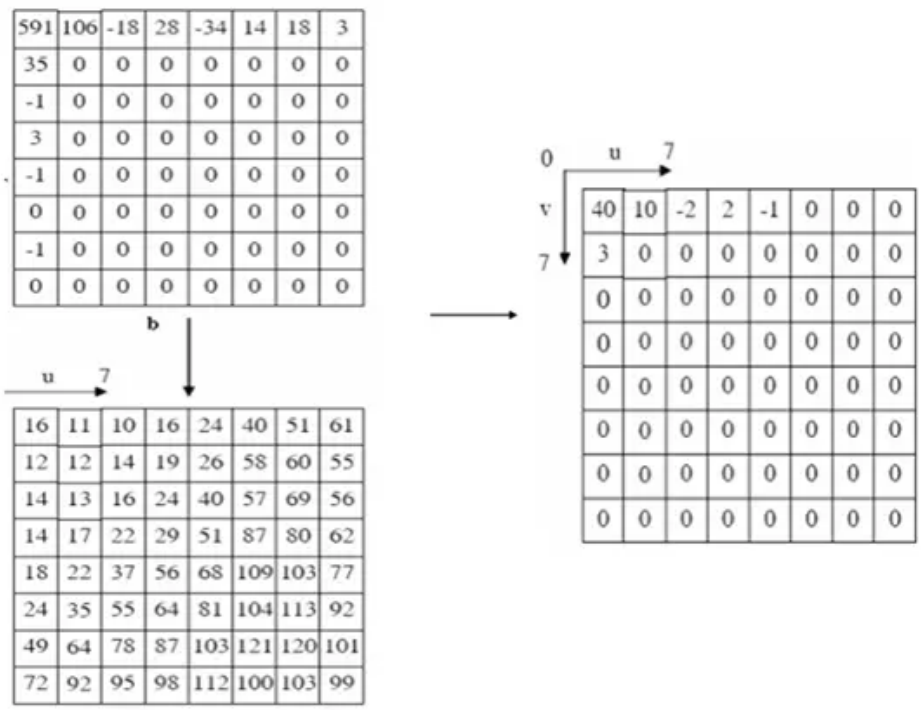
Trong đó:

Fq(u,v): Ma trận kết quả

F(u,v): Ma trận có được từ DCT

Q(u,v): Ma trận lượng tử hóa

Ví dụ:



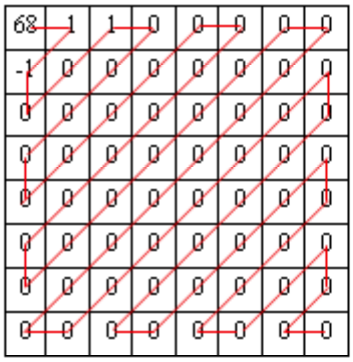
591	106	-18	28	-34	14	18	3
35	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

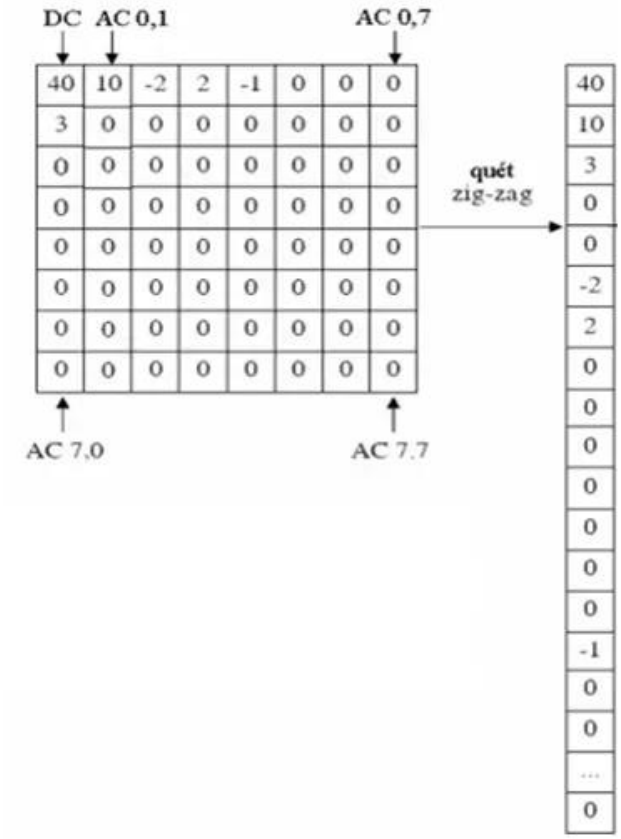
40	10	-2	2	-1	0	0	0
3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Kết quả tìm được (gọi là ma trận lượng tử) được sử dụng thuật toán Zig-Zag để đưa về ma trận 1 chiều được sắp xếp theo thứ tự giảm dần mức độ quan trọng của các hệ số.

Nguyên lý hoạt động của thuật toán Zig-Zag có thể được hình dung như sau:



Ví dụ:

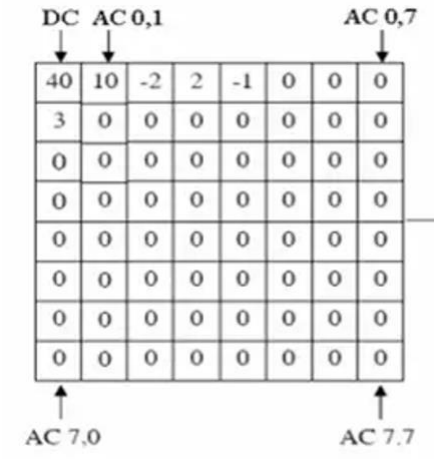


2.2.3.4. MÃ HÓA ENTROPY

Sau khi có mảng 1 chiều lượng tử từ bước lượng tử hóa. Ta tiếp tục thực hiện bước mã hóa Entropy bao gồm mã hóa AC và mã hóa DC.

Hệ số DC nằm ở vị trí đầu tiên của mảng 1 chiều, thể mức độ thay đổi giữa các khối 8x8 liên tiếp.

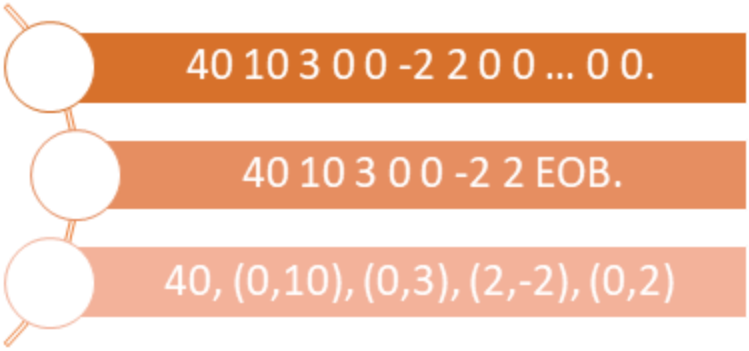
Hệ số AC nằm ở các vị trí còn lại của mảng 1 chiều, biểu diễn sự khác biệt giữa các pixel trong một khối 8x8.



Trước khi bước vào mã hóa DC và mã hóa AC. Ta thực hiện biến đổi đối với ma trận 1 chiều lượng tử.

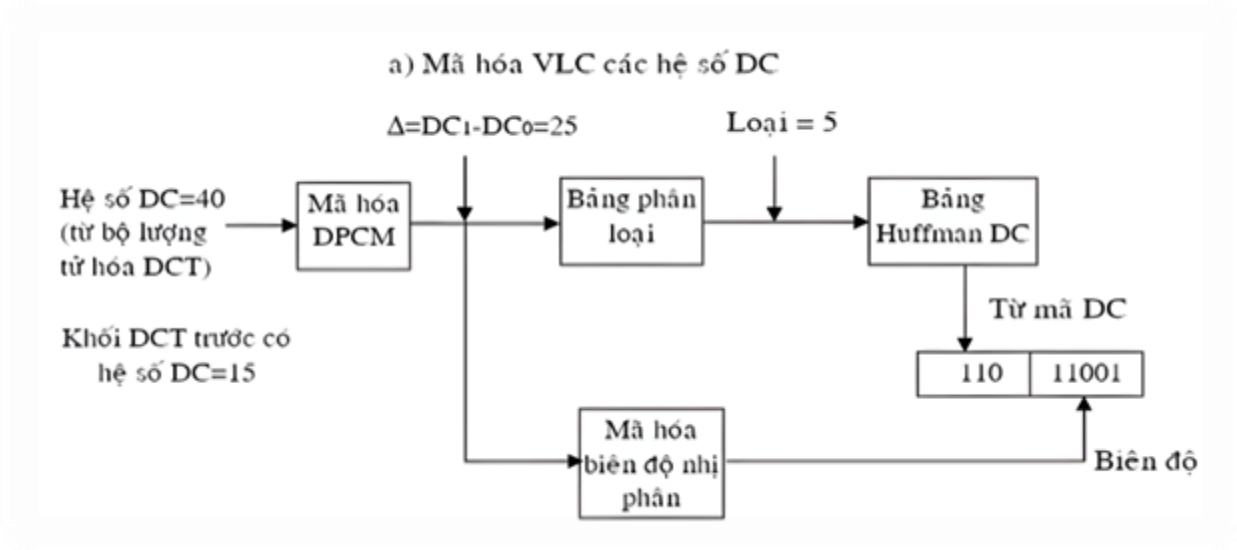
- Nếu các phần tử còn lại của mảng đều là giá trị 0, ta cho mảng kết thúc sớm. Kí hiệu EOB được đặt ở phía sau kí tự cuối cùng khác 0 của mảng.
- Biến đổi RLC đối với các phần tử là hệ số AC trong mảng (Tất cả các phần tử trong mảng ngoại trừ giá trị đầu tiên là hệ số DC). Phép RLC này có tác dụng giảm kích thước của các kí tự 0 liên tiếp trùng nhau.Mỗi phần tử sẽ gồm 2 giá trị:
  - Giá trị thứ nhất là số lượng phần tử có giá trị 0 phía trước phần tử đó.
  - Giá trị thứ hai là giá trị của chính bản thân phần tử đó.

Ví dụ:



a. Mã hóa DC

Các bước mã hóa DC được trình bày một cách tổng quan như sau:



Bảng Huffman có thể được xây dựng mới trong quá trình code hoặc sử dụng bảng Huffman được dựng sẵn như ví dụ dưới đây:

Range	DC Difference Category	AC Category
0	0	N/A
−1, 1	1	1
−3, −2, 2, 3	2	2
−7, . . . , −4, 4, . . . , 7	3	3
−15, . . . , −8, 8, . . . , 15	4	4
−31, . . . , −16, 16, . . . , 31	5	5
−63, . . . , −32, 32, . . . , 63	6	6
−127, . . . , −64, 64, . . . , 127	7	7
−255, . . . , −128, 128, . . . , 255	8	8
−511, . . . , −256, 256, . . . , 511	9	9
−1023, . . . , −512, 512, . . . , 1023	A	A
−2047, . . . , −1024, 1024, . . . , 2047	B	B
−4095, . . . , −2048, 2048, . . . , 4095	C	C
−8191, . . . , −4096, 4096, . . . , 8191	D	D
−16383, . . . , −8192, 8192, . . . , 16383	E	E
−32767, . . . , −16384, 16384, . . . , 32767	F	N/A

Category	Base Code	Length	Category	Base Code	Length
0	010	3	6	1110	10
1	011	4	7	11110	12
2	100	5	8	111110	14
3	00	5	9	1111110	16
4	101	7	A	11111110	18
5	110	8	B	111111110	20

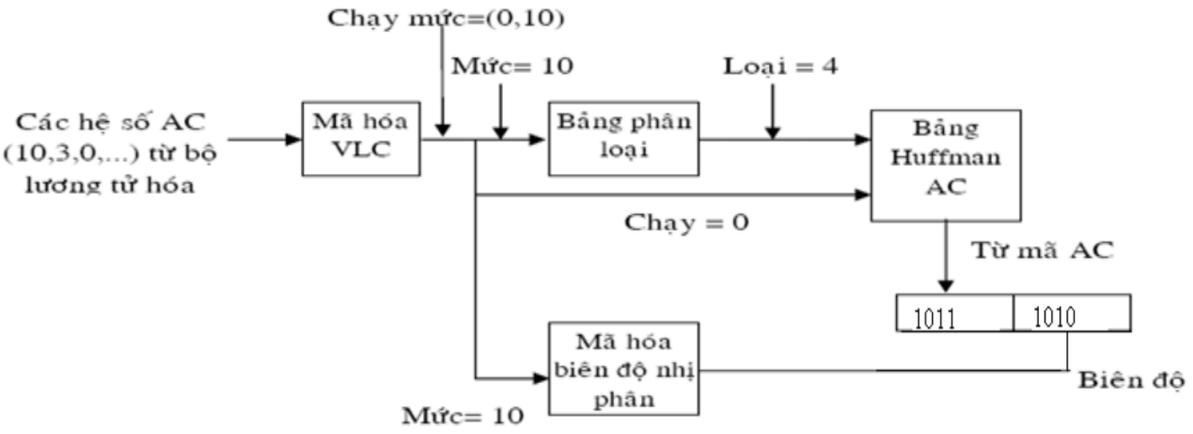
Các bước thực hiện mã hóa DC có thể được hiểu dễ dàng hơn với ví dụ sau đây:

- Giả sử ta có giá trị DC của ma trận hiện hành là 40, giá trị DC của ma trận trước đó là 15.
- Bước 1: $\Delta = DC1 - DC0 = 40 - 15 = 25$
- Bước 2: Tìm category của số vừa tìm được trong bảng DC difference Category. Xét bảng DC difference Category: Số 25 nằm ở vị trí có category là 5 trong bảng. Xét bảng basecode, giá trị này được encode với giá trị là 100. Ta ghi nhận giá trị 100.
- Bước 3: Kiểm tra số vừa tìm được đứng thứ tại vị trí thứ bao nhiêu trong category của nó. Số 25 nằm ở vị trí thứ 25 trong category 5. Số 25 có mã nhị phân là 11001.
- Bước 4: Tổng hợp từ bước 2 và bước 3, ta có:

  Mã hóa DC là: 10011001

  Vậy ta có mã hóa DC là 10011001

b. Mã hóa AC



Mã hóa AC được thực hiện khá tương tự với mã hóa DC như sau:

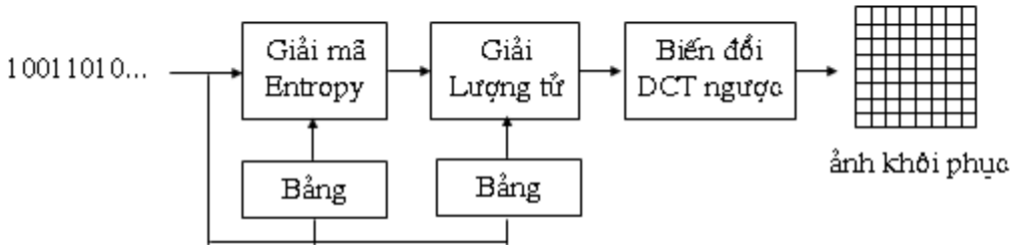
Range	DC Difference Category	AC Category
0	0	N/A
−1, 1	1	1
−3, −2, 2, 3	2	2
−7, . . . , −4, 4, . . . , 7	3	3
−15, . . . , −8, 8, . . . , 15	4	4
−31, . . . , −16, 16, . . . , 31	5	5
−63, . . . , −32, 32, . . . , 63	6	6
−127, . . . , −64, 64, . . . , 127	7	7
−255, . . . , −128, 128, . . . , 255	8	8
−511, . . . , −256, 256, . . . , 511	9	9
−1023, . . . , −512, 512, . . . , 1023	A	A
−2047, . . . , −1024, 1024, . . . , 2047	B	B
−4095, . . . , −2048, 2048, . . . , 4095	C	C
−8191, . . . , −4096, 4096, . . . , 8191	D	D
−16383, . . . , −8192, 8192, . . . , 16383	E	E
−32767, . . . , −16384, 16384, . . . , 32767	F	N/A

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	**1010 (= EOB)**	**4**			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	1111111111000000	17
0/3	100	6	8/3	1111111110110111	19
0/4	1011	8	8/4	1111111110111000	20
0/5	11010	10	8/5	1111111110111001	21
0/6	111000	12	8/6	1111111110111010	22
0/7	1111000	14	8/7	1111111110111011	23
0/8	1111110110	18	8/8	1111111110111100	24
0/9	1111111110000010	25	8/9	1111111110111101	25
0/A	1111111110000011	26	8/A	1111111110111110	26
1/1	1100	5	9/1	11111000	10
1/2	111001	8	9/2	1111111110111111	18
1/3	1111001	10	9/3	1111111111000000	19
1/4	11110110	13	9/4	1111111111000001	20
1/5	1111110110	16	9/5	1111111111000010	21
1/6	1111111110000100	22	9/6	1111111111000011	22
1/7	1111111110000101	23	9/7	1111111111000100	23
1/8	1111111110000110	24	9/8	1111111111000101	24
1/9	1111111110000111	25	9/9	1111111111000110	25
1/A	1111111110001000	26	9/A	1111111111000111	26
2/1	11011	6	A/1	11111001	10
2/2	11111000	10	A/2	1111111111001000	18
2/3	1111110111	13	A/3	1111111111001001	19
2/4	1111111110001001	20	A/4	1111111111001010	20
2/5	1111111110001010	21	A/5	1111111111001011	21
2/6	1111111110001011	22	A/6	1111111111001100	22
2/7	1111111110001100	23	A/7	1111111111001101	23

- Giả sử ta có hệ số AC tại một vị trí là (0,10)
- Bước 1: Xét bảng Table Of JPEG Coefficient Coding Categories: Số 5 nằm ở vị trí có category là 3 trong bảng. Xét bảng basecode, giá trị 0/3 được encode là 100. Ta nhận giá trị 100.
- Bước 2: Số 5 nằm ở vị trí thứ 5 trong category 3, 5 có mã nhị phân là 101. Ta nhận giá trị 101.
- Bước 3: Tổng hợp từ bước 1 và bước 2, ta có: Mã hóa AC là: 100101
- Tiếp tục thực hiện tương tự đối với các hệ số AC còn lại trong mảng.

Sau khi thực hiện các quá trình mã hóa Entropy, dãy nhị phân vừa tìm được sẽ được lưu trữ lại, cùng với các thông tin khác như bảng lượng tử, … để phục vụ cho quá trình giải nén sau này.

2.3. TỔNG QUAN QUY TRÌNH GIẢI NÉN



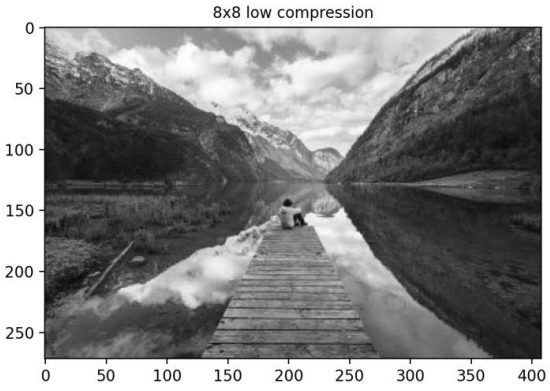
Quá trình giải nén được thực hiện ngược chiều quá trình nén ảnh, từ đoạn mã nhị phân được nén và các thông tin lưu trữ, ảnh được giải nén từ quá trình giải nén Entropy, giải mã lượng tử, biến đổi IDCT, … để đưa ra ảnh khôi phục. Ảnh khôi phục thường bị mất mát dữ liệu so với ảnh gốc ban đầu. Mức độ mất mát phụ thuộc vào độ lớn của ma trận lượng tử cũng như một số bước khác.

PHẦN 3. THỰC NGHIỆM

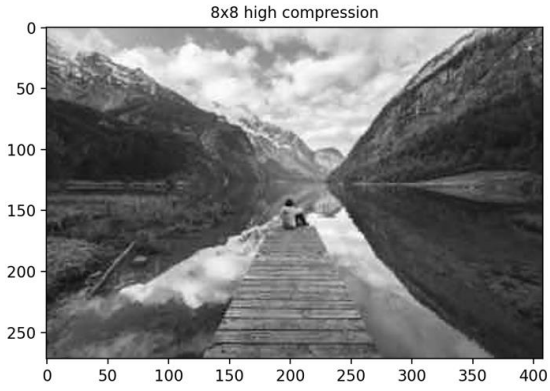
Thuật toán JPEG được áp dụng trên ảnh grayscale, thông qua các bước: trung tâm dữ liệu, DCT, lượng tử hóa, Entropy bằng mã Huffman, …
Kết quả:

Ảnh ban đầu:

- Kích thước ảnh trước khi nén: 887808 bits



225598 bits



90258 bits

Với 8x8 low compression, ảnh ban đầu được nén với ma trận lượng tử:

```
table_8_low = [[1,  1,  1,  1,  1,  2,  2,  4],
               [1,  1,  1,  1,  1,  2,  2,  4],
               [1,  1,  1,  1,  2,  2,  2,  4],
               [1,  1,  1,  1,  2,  2,  4,  8],
               [1,  1,  2,  2,  2,  2,  4,  8],
               [2,  2,  2,  2,  2,  4,  8,  8],
               [2,  2,  2,  4,  4,  8,  8,  16],
               [4,  4,  4,  4,  8,  8, 16,  16]]
```

Với 8x8 high compression, ảnh ban đầu được nén với ma trận lượng tử:

```
table_8_high = [[1,    2,    4,    8,    16,   32,   64,   128],
                [2,    4,    4,    8,    16,   32,   64,   128],
                [4,    4,    8,    16,   32,   64,   128,  128],
                [8,    8,    16,   32,   64,   128,  128,  256],
                [16,   16,   32,   64,   128,  128,  256,  256],
                [32,   32,   64,   128,  128,  256,  256,  256],
                [64,   64,   128,  128,  256,  256,  256,  256],
                [128,  128,  128,  256,  256,  256,  256,  256]]
```

Sự khác biệt về độ lớn của ma trận lượng tử sẽ ảnh hưởng đến chất lượng ảnh sau khi giải nén. Ma trận lượng tử càng lớn, tỉ lệ nén càng cao.

PHẦN 4. KẾT LUẬN

Thuật toán JPEG là một thuật toán vô cùng phổ biến và hữu ích, thường xuyên được sử dụng trong quá trình nén và lưu trữ ảnh kĩ thuật số. Nó cung cấp một sự cân bằng tốt giữa kích thước file và chất lượng ảnh, đồng thời giúp giảm tải băng thông và tăng tốc độ truyền thông ảnh trên mạng.

CHƯƠNG II. THUẬT TOÁN NÉN ẢNH JPEG-2000

PHẦN 1. GIỚI THIỆU

JPEG (Joint Photographic Experts Group) là một định dạng file hình ảnh phổ biến và rộng rãi sử dụng trong công nghệ thông tin và đồ họa. Được tạo ra bởi nhóm chuyên gia trong lĩnh vực hình ảnh (Joint Photographic Experts Group) vào những năm 1990, JPEG nhanh chóng trở thành tiêu chuẩn cho việc nén và lưu trữ hình ảnh số.

Như vậy chúng ta có thể thấy, mặc dù sự ra đời của JPEG mang lại nhiều lợi ích to lớn về nhiều mặt như: Làm giảm nhỏ kích thước ảnh, giảm thời gian truyền và làm giảm chi phí xử lý ảnh với một chất lượng ảnh khá tốt nhưng việc kỹ thuật nén JPEG sẽ làm mất thông tin lúc giải nén (càng nén với hệ số cao thì thông tin càng mất nhiều khi bung) là một hạn chế không nhỏ của phương pháp này. Vì thế để giải quyết vấn đề này, tháng 12/1999 một bản phác thảo tiêu chuẩn nén hình ảnh theo công nghệ mới JPEG2000 - Với tiêu chí dung lượng nhỏ hơn nhưng chất lượng hình ảnh cao hơn đã được ra đời.

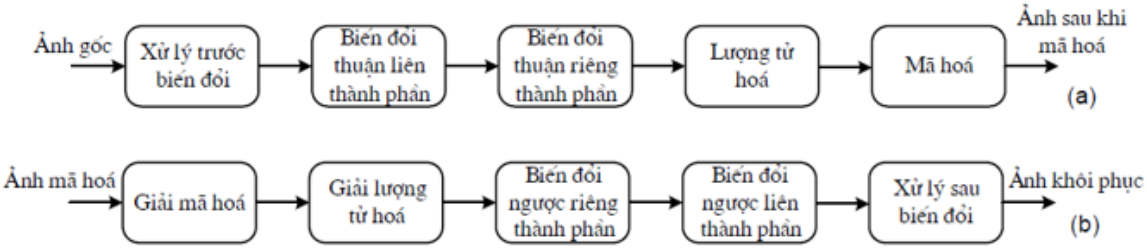
JPEG-2000 là một tiêu chuẩn nén hình ảnh tiên tiến, là phiên bản nâng cấp từ định dạng JPEG truyền thống. Được phát triển bởi nhóm chuyên gia hình ảnh (Joint Photographic Experts Group) vào những năm 2000, JPEG-2000 mang đến những ưu điểm vượt trội và cải tiến so với đối tượng tiền nhiệm của nó. Điểm nổi bật của JPEG-2000 nằm ở khả năng nén hình ảnh với mức độ linh hoạt cao. Điều này cho phép người dùng điều chỉnh chất lượng và độ phân giải của hình ảnh được nén mà không làm mất mát nhiều thông tin. Thậm chí, JPEG-2000 hỗ trợ nén không mất mát (lossless) để đảm bảo việc lưu trữ hình ảnh mà không làm giảm chất lượng.

Ngoài ra, JPEG-2000 hỗ trợ nhiều kênh màu và đa dạng hóa phạm vi màu sắc, cho phép nó được sử dụng rộng rãi trong các lĩnh vực y học, quảng cáo, truyền thông và nhiều ngành công nghiệp khác. Điều này làm cho nó trở thành một công nghệ nén hình ảnh đáng tin cậy và phổ biến trong thế giới số hiện đại. Thuật toán trong kỹ thuật JPEG2000 là chọn một số nhỏ các sóng ngắn, các sóng này được lặp lại ở những nơi khác nhau, tỷ lệ khác nhau đã mô tả chính xác tín hiệu của hình ảnh. File ảnh nén không chứa nhiều hơn số lượng chỉ vị trí và giãn nở của từng sóng ngắn. Việc áp dụng kỹ thuật mã hóa theo từng khối, theo từng khu vực ưu tiên của hình ảnh (ROI -Regional Of Interest) cũng là một sự tiến bộ đáng kể trong thuật toán mã hóa JPEG2000.

PHẦN 2. THUẬT TOÁN

2.1. TỔNG QUAN QUY TRÌNH

Dưới đây là mô hình thể hiện trình tự mã hóa và giải mã JPEG-2000:



Hình 1: Trình tự mã hóa (a) và giải mã (b) JPEG-2000

2.2. CHI TIẾT

2.2.1. TRÌNH TỰ MÃ HÓA JPEG-2000

Bước 1. Xử lý trước biến đổi:

Xử lý trước (preprocessing) là bước quan trọng trong quá trình nén hình ảnh JPEG-2000 để tối ưu hóa chất lượng và hiệu suất nén. Bước này bao gồm một số pha chuẩn bị dữ liệu và cải thiện thông tin hình ảnh trước khi áp dụng biến đổi JPEG-2000. Do sử dụng biến đổi Wavelet, JPEG2000 cần có dữ liệu ảnh đầu vào ở dạng đối xứng qua 0.

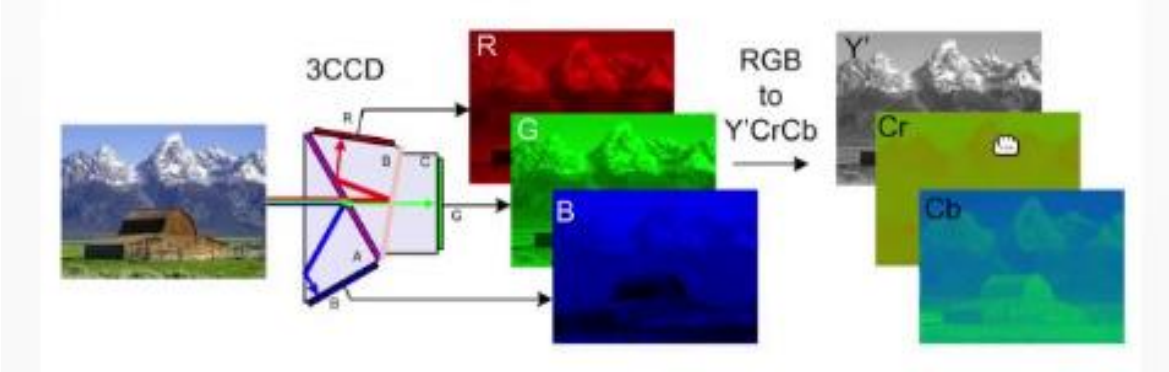
Bước 2. Biến đổi liên thành phần:

Biến đổi liên thành phần (Joint Component Transform) là một trong những bước quan trọng trong quá trình nén hình ảnh theo chuẩn JPEG-2000. Bước này thực hiện sau khi đã thực hiện xử lý trước dữ liệu ảnh và tách thành phần tần số sử dụng biến đổi wavelet (DWT). Mục tiêu của biến đổi liên thành phần là cải thiện tính tương quan giữa các thành phần (chẳng hạn như các kênh màu) của ảnh trước khi tiến hành mã hóa và lưu trữ chúng. Giai đoạn này sẽ loại bỏ tính tương quan giữa các thành phần của ảnh. JPEG2000 sử dụng hai loại biến đổi liên thành phần là biến đổi màu thuận nghịch (Reversible Color Transform - RCT) và biến đổi màu không thuận nghịch (Irreversible Color Transform - ICT) trong đó biến đổi thuận

nghịch làm việc với các giá trị nguyên, còn biến đổi không thuận nghịch làm việc với các giá trị thực.

ICT và RCT chuyển dữ liệu ảnh từ không gian màu RGB sang YCrCb. RCT được áp dụng trong cả hai dạng thức nén có tổn thất và không tổn thất, còn ICT chỉ áp dụng cho nén có tổn thất. Việc áp dụng các biến đổi này trước khi nén ảnh không nằm ngoài mục đích làm tăng hiệu quả nén. Các thành phần Cr, Cb có ảnh hưởng rất ít tới sự cảm nhận hình ảnh của mắt trong khi thành phần độ chói Y có ảnh hưởng rất lớn tới ảnh.

$$
\begin{aligned}
Y' &= 16 + \frac{65.738 \cdot R'_D}{256} + \frac{129.057 \cdot G'_D}{256} + \frac{25.064 \cdot B'_D}{256} \\
C_B &= 128 - \frac{37.945 \cdot R'_D}{256} - \frac{74.494 \cdot G'_D}{256} + \frac{112.439 \cdot B'_D}{256} \\
C_R &= 128 + \frac{112.439 \cdot R'_D}{256} - \frac{94.154 \cdot G'_D}{256} - \frac{18.285 \cdot B'_D}{256}
\end{aligned}
$$



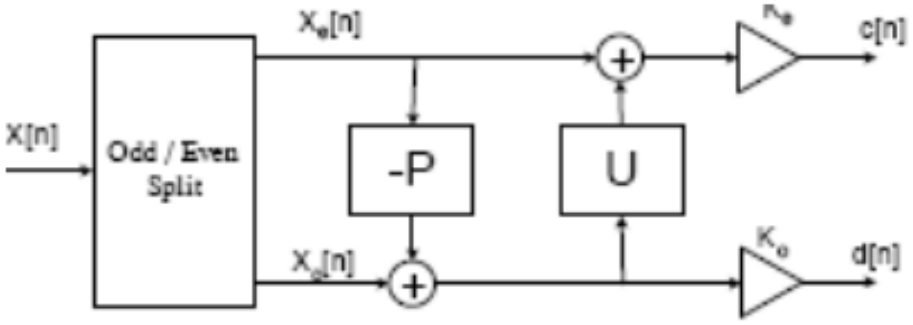
Hình 2: Chuyển đổi từ không gian màu RGB sang YcrCb

Bước 3. Biến đổi riêng thành phần:

Biến đổi riêng thành phần (Component Separation Transform) là một bước quan trọng trong quá trình nén hình ảnh theo chuẩn JPEG-2000. Bước này thực hiện sau khi đã thực hiện xử lý trước dữ liệu ảnh và tách thành phần tần số sử dụng biến đổi wavelet (DWT) và biến đổi liên thành phần. Mục tiêu của biến đổi riêng

thành phần là tách các thành phần (chẳng hạn như các kênh màu) ra khỏi ảnh để được nén riêng biệt, từ đó cải thiện hiệu suất nén và chất lượng hình ảnh.

Biến đổi riêng thành phần được áp dụng trong JPEG2000 chính là biến đổi Wavelet. Để đảm bảo tính toàn vẹn thông tin cũng phải áp dụng các phép biến đổi thuận nghịch hoặc bất thuận nghịch. Do phép biến đổi Wavelet không phải là một phép biến đổi trực giao như biến đổi DCT mà là một phép biến đổi băng con nên các thành phần sẽ được phân chia thành các băng tần số khác nhau và mỗi băng sẽ được mã hóa riêng rẽ. JPEG2000 áp dụng biến đổi Wavelet nguyên thuận nghịch 5/3 (IWT) và biến đổi thực không thuận nghịch Daubechies 9/7. Việc tính toán biến đổi trong JPEG2000 này sẽ được thực hiện theo phương pháp Lifting. Do biến đổi Wavelet 5/3 là biến đổi thuận nghịch nên có thể áp dụng cho nén ảnh theo cả hai phương pháp, có tổn thất và không tổn thất trong khi biến đổi 9/7 chỉ áp dụng cho nén ảnh theo phương pháp có tổn thất thông tin.



Hình 3: Mã hóa băng tần số khác nhau

Bước 4. Lượng tử hóa:

Các hệ số của phép biến đổi sẽ được tiến hành lượng tử hoá. Quá trình lượng tử hoá cho phép đạt tỷ lệ nén cao hơn bằng cách thể hiện các giá trị biến đổi với độ chính xác tương ứng cần thiết với mức chi tiết của ảnh cần nén. Các hệ số biến đổi

sẽ được lượng tử hoá theo phép lượng tử hoá vô hướng. Các hàm lượng tử hoá khác nhau sẽ được áp dụng cho các băng con khác nhau và được thực theo biểu thức sau với Δ là bước lượng tử, U (x, y) là giá trị băng con đầu vào; V (x, y) là giá trị sau lượng tử hoá:

$$V(x,y) = \left\lfloor \left| U(x,y) \right| / \Delta \right\rfloor \operatorname{sgn} U(x,y)$$

Hình 4: Công thức lượng tử hóa

Bước 5. Mã hóa:

Trong thực tế các phương pháp mã hoá ảnh được áp dụng khi nén ảnh bằng biến đổi Wavelet cũng như JPEG2000 thì có hai phương pháp được coi là cơ sở và được áp dụng nhiều nhất: phương pháp SPIHT và phương pháp EZW.

Phương pháp mã hoá SPIHT:

SPIHT (Set Partitioning in Hierarchical Trees) là một phương pháp mã hóa dữ liệu được sử dụng trong quá trình nén hình ảnh theo chuẩn JPEG-2000. Phương pháp này được phát triển bởi Amir Said và William A. Pearlman vào năm 1996 và sau đó đã được sử dụng rộng rãi trong nén hình ảnh và video.

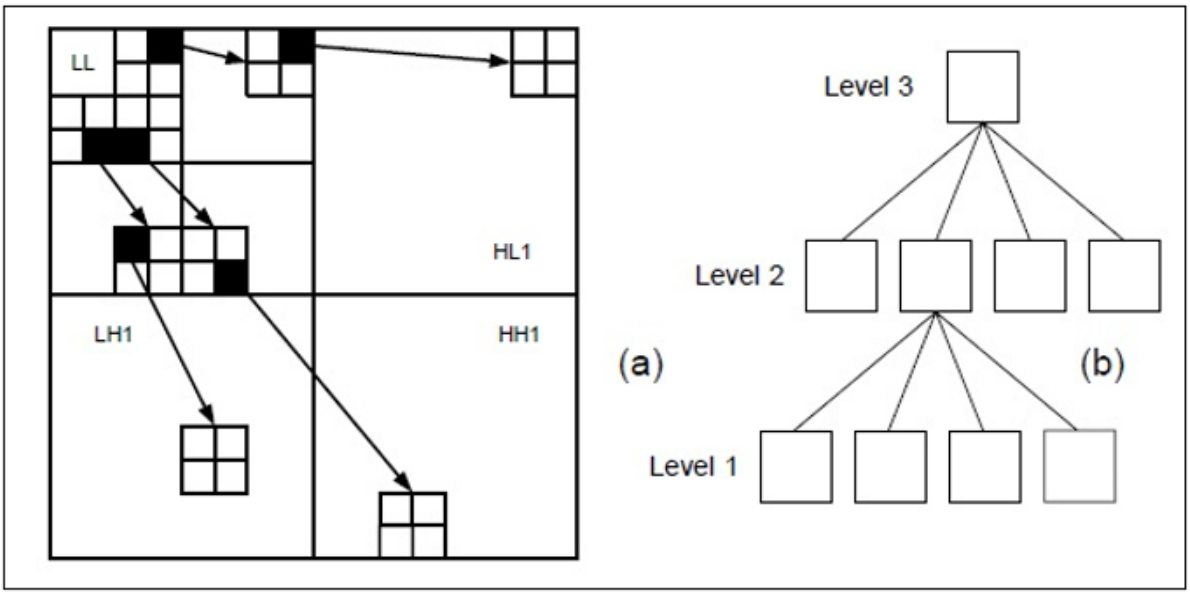
Nguyên lý hoạt động của SPIHT là sử dụng mô hình cây phân cấp để mã hóa các hệ số biến đổi (ví dụ: DWT) được tạo ra từ quá trình biến đổi liên thành phần và riêng thành phần. Phương pháp này tận dụng tính tương quan giữa các hệ số biến đổi để giảm số lượng dữ liệu cần được lưu trữ và truyền tải, đồng thời vẫn giữ được chất lượng hình ảnh tốt. phương pháp này sử dụng kỹ thuật embedded coding; điều đó có nghĩa là một ảnh sau nén với kích cỡ (lưu trữ) lớn (tỷ lệ nén thấp) sẽ chứa chính dữ liệu sau nén của ảnh có kích cỡ (lưu trữ) nhỏ (tỷ lệ nén

cao). Bộ mã hoá chỉ cần nén một lần nhưng có thể giải nén ra nhiều mức chất lượng khác nhau.

Phương pháp mã hóa EZW:

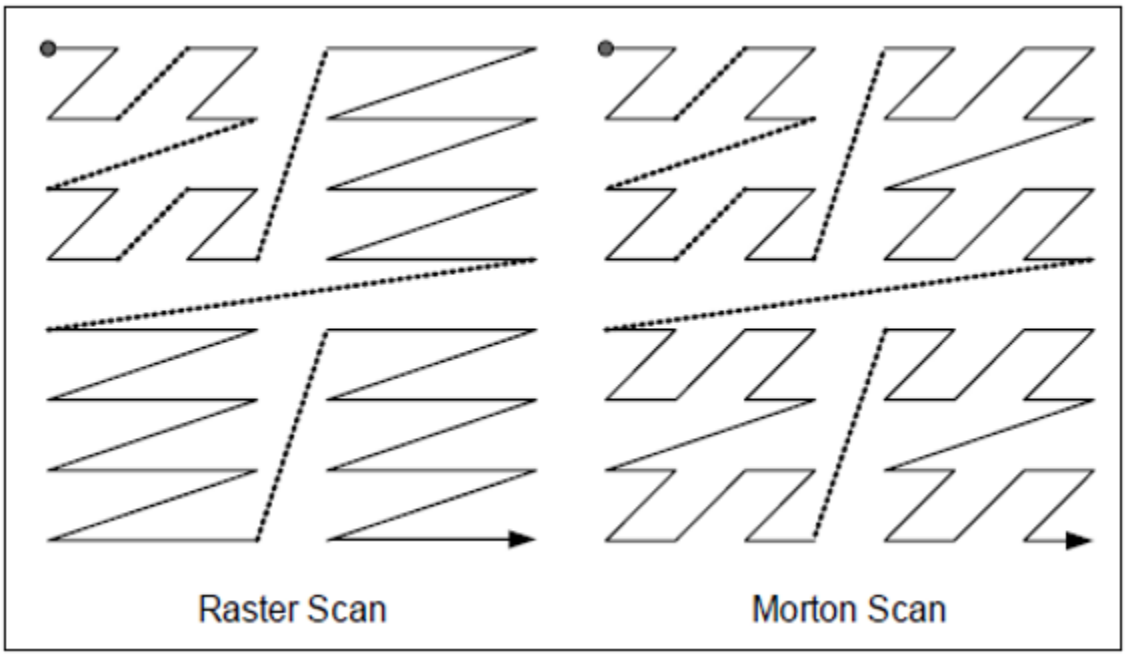
Phương pháp mã hoá EZW (Embedded Zerotree Wavelet Encoder) cũng dựa trên cơ sở phép mã hoá luỹ tiến (progressive coding) giống như phương pháp mã hoá SPIHT. Phương pháp này chủ yếu dựa trên khái niệm về cây zero (zerotree). Về cơ bản, thuật toán này dựa trên hai nguyên tắc như đã trình bày ở phần phương pháp mã hoá SPIHT. Sau đây chúng ta sẽ xem xét các khái niệm cơ bản của thuật toán.

Cây tứ phân: Sau khi áp dụng biến đổi Wavelet ứng với các mức phân giải khác nhau chúng ta có thể biểu diễn các hệ số biến đổi dưới dạng một cây. Ta thấy rằng với cây biểu diễn này cứ mỗi nút cha thì có 4 nút con. Kết quả này là do quá trình biến đổi Wavelet ở các tỷ lệ khác nhau. Các cây này được gọi là các cây tứ phân (quadtree). Sơ đồ cây tứ phân:



Hình 5: Sơ đồ cây tứ phân (a) và sự phân mức (b)

Cây zero (zerotree): Cây zero là một cây tứ phân, trong đó tất cả các nút của nó đều nhỏ hơn nút gốc. Một cây như vậy khi mã hoá sẽ được mã hoá bằng một đối tượng duy nhất và khi giải mã thì chúng ta cho tất cả các giá trị bằng không. Ngoài ra để có thể mã hoá được các hệ số Wavelet trong trường hợp này, giá trị của nút gốc phải nhỏ hơn giá trị ngưỡng đang được xem xét ứng với hệ số Wavelet đó Sau khi có đủ các khái niệm cần thiết về cây tứ phân và cây zero, chúng ta có thể trình bày nguyên lý hoạt động của thuật toán. Thuật toán sẽ mã hoá các hệ số theo thứ tự giảm dần. Chúng ta sẽ dùng một giá trị gọi là ngưỡng (threshold) và sử dụng ngưỡng này để tiến hành mã hoá các hệ số biến đổi. Các hệ số được mã hoá theo thứ tự từ vùng tần số thấp đến vùng tần số cao. Và chỉ những hệ số có giá trị tuyệt đối lớn hơn hoặc bằng ngưỡng thì mới được mã hoá. Tiếp theo giảm ngưỡng và tiếp tục làm như vậy cho tới khi ngưỡng đạt tới một giá trị nhỏ hơn giá trị của hệ số nhỏ nhất.



Hình 6: Hai cách sắp xếp thứ tự các hệ số biến đổi

2.2.2. TRÌNH TỰ GIẢI MÃ JPEG-2000

Ta có thể hiểu quá trình giải mã JPEG-2000 là một quá trình đảo ngược lại của quá trình mã hóa JPEG-2000. Củ thể ở từng bước được đảo ngược như sau:

Bước 1. Giải mã hóa:

Giải mã hóa bằng SPIHT (Set Partitioning in Hierarchical Trees): Giải nén dữ liệu hệ số: Đầu tiên, ta giải nén dữ liệu hệ số đã được lưu trữ bằng phương pháp SPIHT trong quá trình nén. Bước này giải nén các hệ số theo từng bit-plane và tạo lại các hệ số biến đổi cho ảnh. Giải nén liên thành phần và riêng thành phần: Tiếp theo, ta giải nén liên thành phần (Joint Component Decoding) và giải nén riêng thành phần (Component Separation Decoding) để tạo lại các thành phần của ảnh, chẳng hạn như các kênh màu Y, Cb, Cr đối với ảnh màu. Giải nén biến đổi: Sau đó, ta giải nén các hệ số biến đổi đã được tạo ra từ quá trình biến đổi liên thành phần và riêng thành phần. Bước này giúp khôi phục thông tin chi tiết của ảnh. Kết hợp các thành phần: Cuối cùng, ta kết hợp lại các thành phần đã được giải nén để tái tạo lại ảnh gốc.

Giải mã hóa bằng EZW (Embedded Zerotree Wavelet): Giải nén dữ liệu hệ số: Tương tự như phương pháp SPIHT, ta giải nén dữ liệu hệ số đã được lưu trữ bằng phương pháp EZW trong quá trình nén. Các hệ số được giải nén dựa trên các thông tin thứ tự và mức độ tầm quan trọng. Giải nén biến đổi: Sau đó, ta giải nén các hệ số biến đổi đã được tạo ra từ quá trình biến đổi liên thành phần và riêng thành phần, giúp khôi phục thông tin chi tiết của ảnh. Kết hợp các thành phần: Cuối cùng, ta kết hợp lại các thành phần đã được giải nén để tái tạo lại ảnh gốc.

Cả hai phương pháp SPIHT và EZW đều giúp tái tạo lại chất lượng hình ảnh ban đầu sau khi đã trải qua các bước mã hóa và nén. Tuy nhiên, có sự khác biệt

nhất định về cách mã hóa dữ liệu và cơ chế giải mã hóa, nhằm đạt được hiệu suất nén và chất lượng hình ảnh tối ưu.

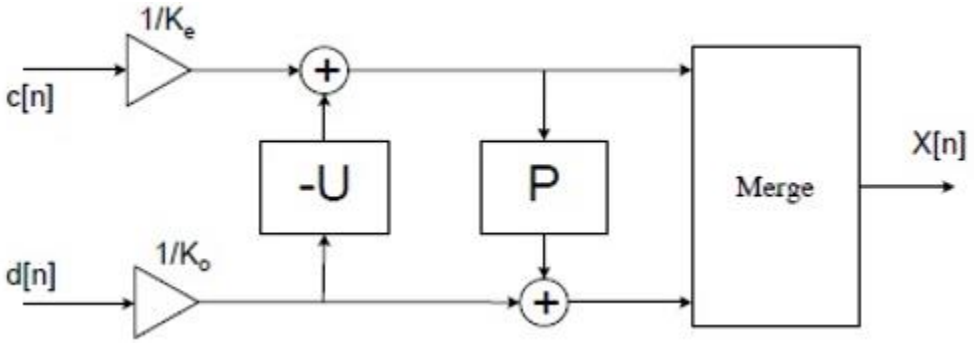
Bước 2. Giải lượng tử hóa:

Bước lượng tử của mỗi băng do đó phải có ở trong dòng bít truyền đi để phía thu có thể giải lượng tử cho ảnh. Công thức giải lượng tử hoá là:

$$U(x,y)=\left[V(x,y)+r\,\mathrm{sgn}\,V(x,y)\right]\Delta$$

Hình 7: Công thức giải lượng tử hóa

Bước 3. Biến đổi ngược riêng thành phần:



Hình 8: Quá trình biến đổi ngược liên thành phần

Bước 4. Biến đổi ngược liên thành phần:

Biến đổi ngược liên thành phần (Inverse Joint Component Transform) là một trong những bước quan trọng trong quá trình giải nén ảnh JPEG-2000, nó được thực hiện sau khi đã giải nén dữ liệu hệ số và các bước giải mã hóa khác. Bước này thực hiện việc đảo ngược quá trình biến đổi liên thành phần (Joint Component Transform) đã thực hiện trong quá trình nén để tái tạo lại các thành phần của ảnh (chẳng hạn như các kênh màu Y, Cb, Cr đối với ảnh màu) ban đầu.

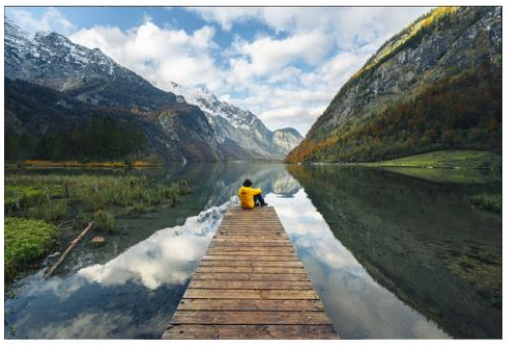
Bước biến đổi ngược liên thành phần là quan trọng để tái tạo lại các thành phần của ảnh ban đầu và khôi phục chất lượng hình ảnh gần giống như ảnh trước khi thực hiện quá trình nén JPEG-2000.

Bước 5. Xử lý sau biến đổi:

Giai đoạn xử lý sau biến đổi sẽ trả lại giá trị gốc ban đầu cho dữ liệu ảnh.

PHẦN 3. THỰC NGHIỆM

Thử nghiệm trên một tấm ảnh có kích thước 2040x1360 cho ra dung lượng sau khi nén được cải thiện rõ rệt.



size = 519 kb

size = 109 kb

PHẦN 4. SO SÁNH GIỮA JPEG VÀ JPEG-2000

Một tính năng quan trọng và là ưu điểm rõ nét nhất của JPEG2000 so với JPEG cũng như các chuẩn nén ảnh khác như MPEG 4 VTC hay JPEG - LS v. v.... là JPEG2000 đưa ra cả hai kỹ thuật nén có tổn thất và không tổn thất theo cùng một cơ chế mã hoá nghĩa là JPEG2000 thực hiện tất cả các dạng thức của JPEG chỉ bằng một cơ chế mã hoá duy nhất.

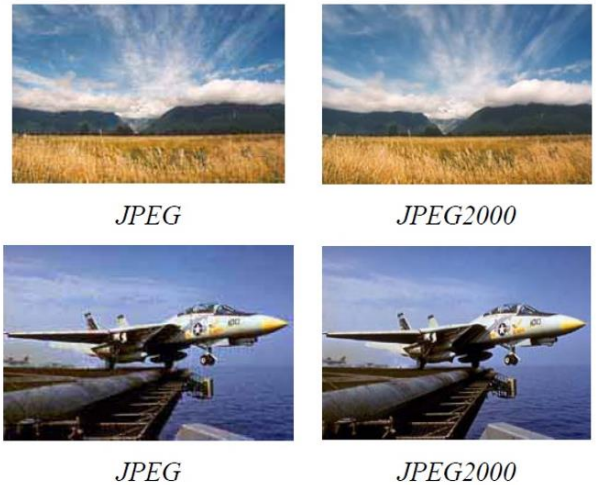
Tính năng ưu việt thứ hai của JPEG2000 so với JPEG chính là trong dạng thức nén có tổn thất thông tin, JPEG2000 có thể đưa ra tỷ lệ nén cao hơn nhiều so với JPEG.

Theo công thức tính *PSNR* trong đơn vị dB, chúng ta có: (*b* là số bít dùng biểu diễn một *pixel* trên ảnh gốc).

$$\mathrm{PSNR(dB)} = -20\log\left(\frac{\mathrm{RMSE}}{2^b - 1}\right)$$

Hình 9: Công thức PSNR

Áp dụng cho các tấm ảnh JPEG và JPEG-2000:



Hình 10: So sánh JPEG và JPEG-2000 trên PSNR

Bit per pixel	0. 125	0. 50	2.00
Ảnh 1 theo *JPEG*	24.42	31.17	35. 15
Ảnh 1 theo *JPEG2000*	28. 12	32.95	37. 35
Ảnh 2 theo *JPEG*	22.6	28. 92	35. 99
Ảnh 2 theo *JPEG2000*	24.85	31.13	38. 80

Hình 11: Kết quả so sánh

Tính năng ưu việt thứ 3 của JPEG2000 so với JPEG là chuẩn nén ảnh này có thể hiển thị được các ảnh với độ phân giải và kích thước khác nhau từ cùng một ảnh nén.

PHẦN 5. KẾT LUẬN

Tóm lại, kĩ thuật JPEG-2000 là một công nghệ nén hình ảnh tiên tiến và đa dạng, cung cấp hiệu suất nén cao, chất lượng hình ảnh tốt và tính tin cậy trong quá trình truyền tải và lưu trữ. Tuy nhiên, việc tính toán phức tạp có thể khiến nó không thích hợp cho các ứng dụng có tài nguyên hạn chế.

CHƯƠNG III. THUẬT TOÁN NÉN ẢNH PNG

PHẦN 1. GIỚI THIỆU

1.1. ĐẶT VẤN ĐỀ

Định dạng hình ảnh PNG (Portable Network Graphics) là một định dạng nén không mất mát phổ biến và được sử dụng rộng rãi trên web và trong các ứng dụng khác. PNG được phát triển nhằm thay thế cho định dạng GIF và định dạng nén có mất mát trong việc lưu trữ và chia sẻ hình ảnh trực tuyến.

PNG mang lại một số lợi ích đáng kể khi sử dụng trong việc nén ảnh. Một trong những lợi ích quan trọng nhất là khả năng giữ nguyên chất lượng hình ảnh ban đầu mà không gây mất mát thông tin. Điều này đảm bảo rằng hình ảnh sau khi nén vẫn giữ được độ phân giải, màu sắc và chi tiết gốc, đáp ứng yêu cầu của các ứng dụng đòi hỏi chất lượng hình ảnh cao.

Một đặc điểm nổi bật của PNG là khả năng hỗ trợ hình ảnh trong suốt. Điều này cho phép người dùng tạo ra hình ảnh có phần nền trong suốt, thích hợp cho các ứng dụng như logo, biểu tượng hay hiệu ứng đồ hoạ đặc biệt. PNG cũng hỗ trợ các lớp kênh trong suốt, cho phép kết hợp hình ảnh với nền và tạo ra hiệu ứng đặc biệt khi hiển thị trên các trình duyệt và ứng dụng.

Định dạng PNG không giới hạn số lượng màu sắc, bao gồm cả hình ảnh xám, hình ảnh chỉnh màu và hình ảnh RGBA. Điều này cho phép tái tạo chính xác các màu sắc gốc của hình ảnh và đáp ứng nhu cầu đa dạng về hiển thị màu sắc trong các ứng dụng.

Ngoài ra, PNG cũng giúp giảm kích thước tệp ảnh một cách hiệu quả. Mặc dù không hiệu quả bằng các định dạng nén có mất mát như JPEG, PNG vẫn có khả năng giảm kích thước tệp ảnh mà không làm mất thông tin. Điều này giúp giảm

băng thông cần thiết để tải hình ảnh và tăng tốc độ tải trang, cải thiện trải nghiệm người dùng và tiết kiệm dung lượng lưu trữ.

- Input của thuật toán nén ảnh PNG là một hình ảnh trong bất kỳ định dạng nào (JPEG, BMP, GIF, etc.)
- Output của thuật toán nén ảnh PNG là một tệp hình ảnh được nén theo định dạng PNG. Tệp hình ảnh PNG sẽ có kích thước nhỏ hơn so với tệp gốc nhưng vẫn giữ nguyên chất lượng hình ảnh ban đầu mà không gây mất mát thông tin.

1.2. THÁCH THỨC BÀI TOÁN

Mặc dù định dạng hình ảnh PNG mang lại nhiều lợi ích, nhưng nó cũng đối diện với một số thách thức. Một trong những thách thức chính là kích thước tệp lớn. PNG có thể tạo ra các tệp lớn hơn so với các định dạng nén có mất mát khác, đặc biệt là đối với hình ảnh có nhiều chi tiết và màu sắc phức tạp. Điều này có thể gây khó khăn khi truyền tải và lưu trữ hình ảnh.

Một thách thức khác của PNG là thời gian nén. Do thuật toán nén PNG có độ phức tạp cao, quá trình nén có thể tốn nhiều thời gian, đặc biệt là với các hình ảnh có kích thước lớn. Điều này có thể ảnh hưởng đến hiệu suất và thời gian xử lý của ứng dụng.

Ngoài ra, PNG không phải là lựa chọn hiệu quả cho các hình ảnh có màu sắc phức tạp, như các hình ảnh có gradient màu hoặc màu sắc thay đổi mượt. Các định dạng nén có mất mát như JPEG thường mang lại kết quả kích thước nhỏ hơn cho các loại hình ảnh này và PNG không hỗ trợ hình ảnh động, điều này là một hạn chế so với các định dạng khác như GIF hoặc WebP. Nếu cần lưu trữ hình ảnh động, các định dạng khác sẽ phù hợp hơn.

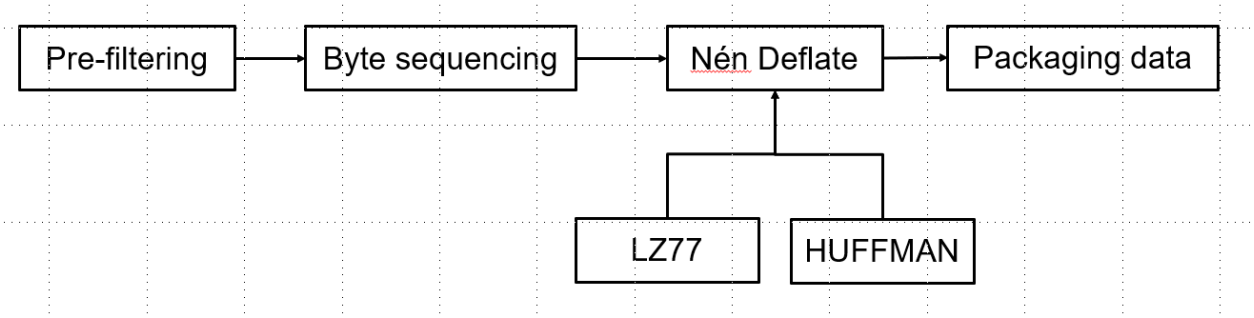
Mặc dù đối mặt với những thách thức này, PNG vẫn được sử dụng rộng rãi trong nhiều trường hợp, nhất là khi yêu cầu bảo tồn chất lượng hình ảnh cao và không chấp nhận mất mát thông tin. Lựa chọn định dạng nén phù hợp sẽ phụ thuộc vào yêu cầu cụ thể của dự án và tình huống sử dụng.

1.3. MỤC TIÊU VÀ PHẠM VI

- Mục tiêu chung ở phần này là xây dựng được các bước của thuật toán nén ảnh PNG. Sử dụng các thư viện có sẵn và dữ liệu để tiến hành đánh giá thuật toán PNG.
- Trong phần này chúng tôi giới hạn phạm vi của đề tài : Dữ liệu chúng tôi sử dụng trong bài toán này là dữ liệu dạng ảnh

PHẦN 2. THUẬT TOÁN

2.1. TỔNG QUAN



Thuật toán:

- Bước 1: Filtering: Trước khi nén, các dòng của ảnh được bộ lọc để giảm sự dự đoán của các pixel kế tiếp dựa trên các pixel trước đó. Có năm kiểu bộ lọc mà PNG sử dụng: None (không bộ lọc), Sub, Up, Average, và Paeth.
- Bước 2: Byte Sequencing: Sau khi được bộ lọc, dữ liệu của hình ảnh được tổ chức thành chuỗi byte.
- Bước 3: Nén Deflate: Chuỗi byte sau cùng sau khi được bộ lọc được nén sử dụng thuật toán nén Deflate. Thuật toán Deflate kết hợp LZ77 và Huffman Coding.
- Bước 4: Packaging the Data: Cuối cùng, dữ liệu nén được đóng gói vào định dạng PNG bao gồm các thông tin như kích thước ảnh, dữ liệu màu sắc và các thông tin khác.

2.2. CHI TIẾT

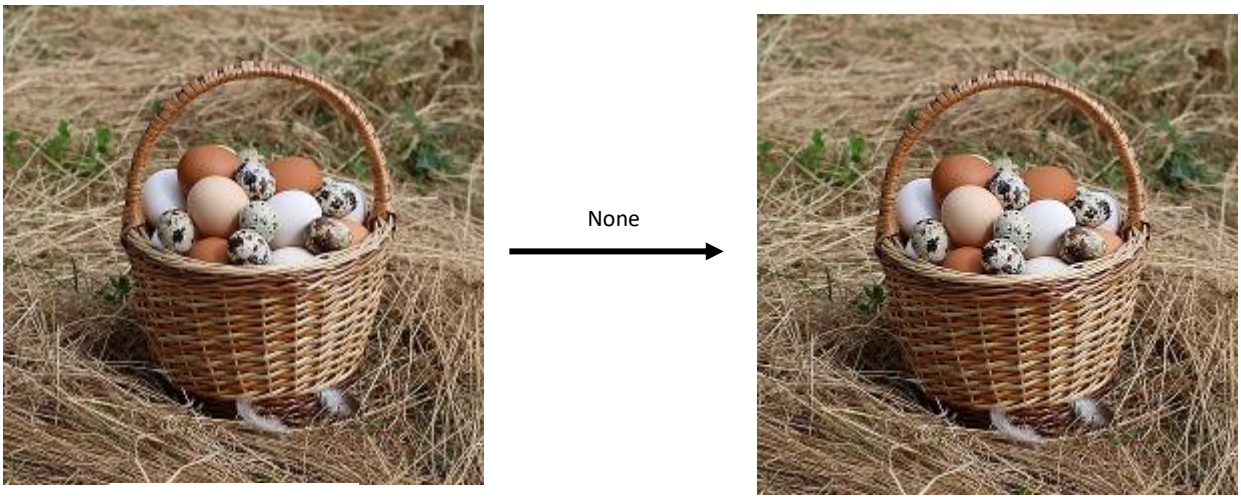
2.2.1. FILTERING

Là quá trình tiền xử lý dữ liệu trước khi nén. Các bộ lọc hoạt động bằng cách thay đổi các giá trị pixel cụ thể dựa trên các giá trị pixel xung quanh nó, nhằm tạo ra sự thay đổi nhỏ và đồng nhất giữa các giá trị pixel liên tiếp. Cách hoạt động giúp số byte cần phải lưu trữ được giảm đi và đồng thời bộ lọc giúp tăng cường

hiệu suất của thuật toán nén (Deflate) bằng cách biến đổi dữ liệu ảnh để tạo ra nhiều chuỗi byte giống nhau hơn, điều này làm tăng khả năng tìm thấy và thay thế các chuỗi lặp lại trong quá trình nén.

- Có năm kiểu bộ lọc mà PNG sử dụng:
    - None: Không áp dụng bất kỳ bộ lọc nào. Dữ liệu pixel giữ nguyên.

      Ví dụ:

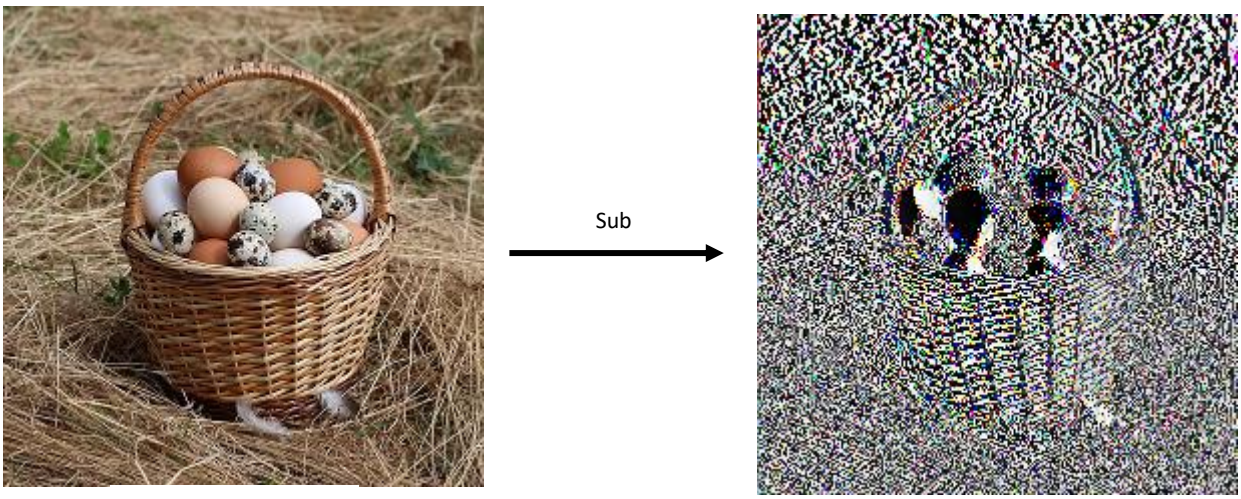


Original

None filtered

    - Sub: Giá trị của mỗi pixel bị trừ bằng giá trị của pixel liền trước nó (theo chiều ngang).

      Ví dụ:



Original

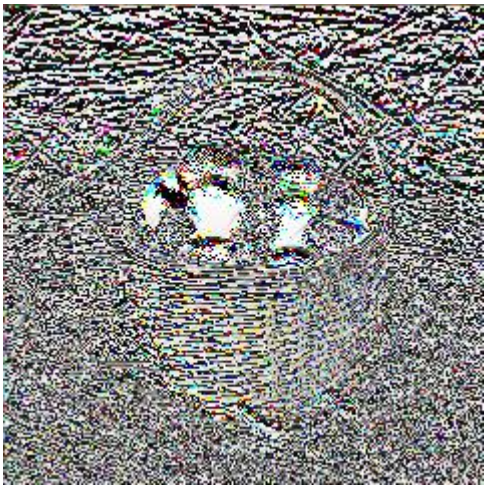
Sub filtered

- Up: Giá trị của mỗi pixel bị trừ bằng giá trị của pixel liền trên nó (theo chiều dọc).

  Ví dụ:

Original

Up



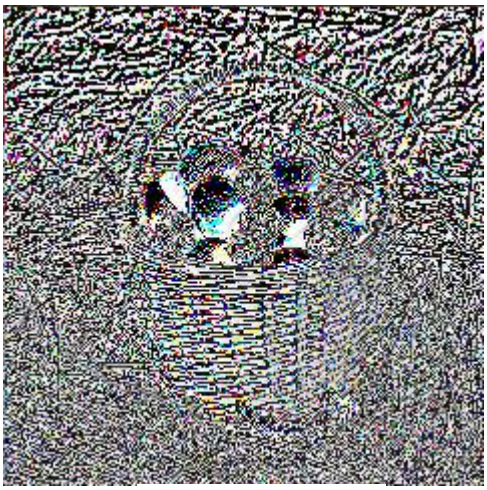
Up filtered

- Average: Giá trị của mỗi pixel bị trừ bằng trung bình của pixel liền trước nó và pixel liền trên nó.

- Ví dụ:

Original

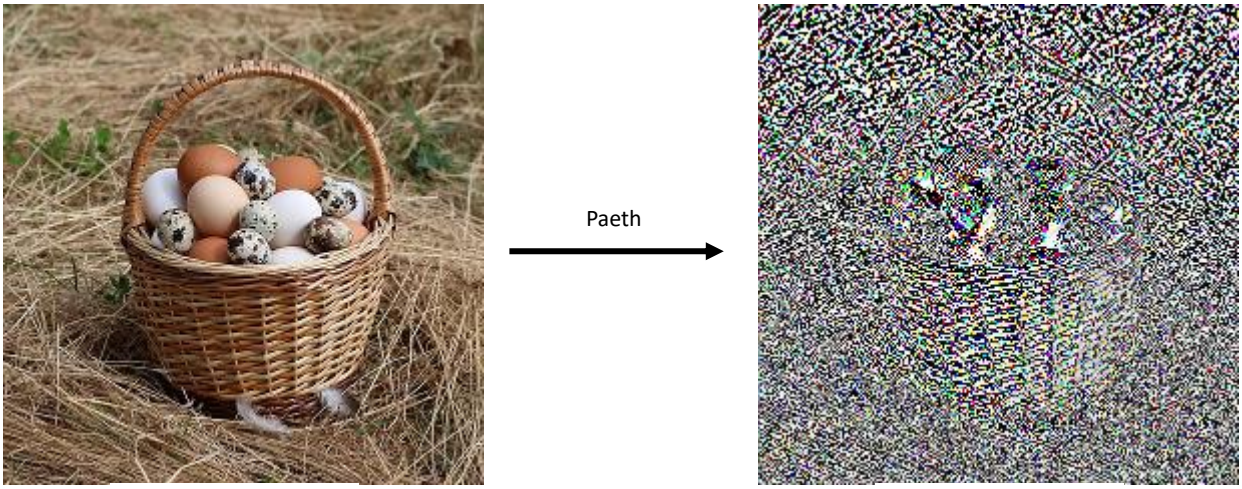
Average



Average filtered

- Paeth: Một loại bộ lọc dự đoán phức tạp hơn. Nó dự đoán giá trị của mỗi pixel dựa trên pixel liền trước nó, pixel liền trên nó và pixel trái trên nó.

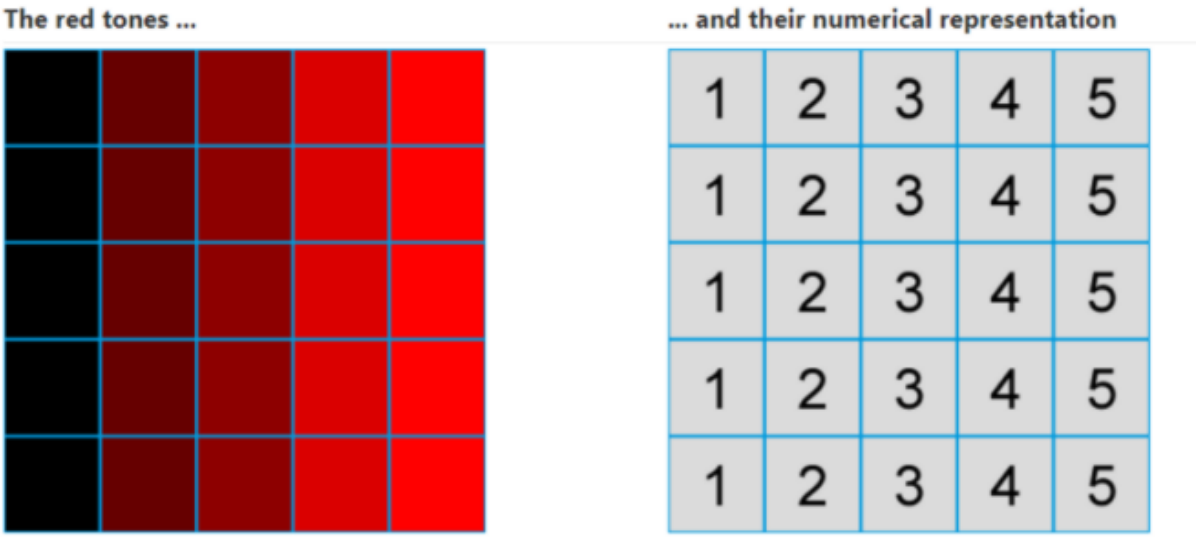
- Ví dụ:



Original

Paeth filtered

- Cụ thể hơn :



Ví dụ trên bên trái là một khối ảnh gốc, sau đó ta sẽ thay các màu sắc đó bằng các giá trị màu tương ứng để tiến hành áp dụng bộ lọc. Sử dụng bộ lọc Sub. Tức là tại giá trị ô hiện tại mới = giá trị ô hiện tại cũ – giá trị ô liền kề trái.

Và ta thu được kết quả như sau:

The values filtered by "Sub"

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Qua đó cho ta thấy. Khi áp dụng một số filter sẽ giúp giảm số byte lữu trữ (ở ví dụ trên từ 3 bit xuống còn 1 bit). Và tạo ra nhiều chuỗi byte giống nhau hơn, điều này làm tăng khả năng tìm thấy và thay thế các chuỗi lặp lại trong quá trình nén.

2.2.2. THUẬT TOÁN NÉN DEFLATE

Thuật toán Deflate là thuật toán nén không mất mát, được áp dụng trong định dạng tệp như ZIP và PNG. Nó kết hợp hai phương pháp: mã hóa Huffman và LZ77. Mã hóa Huffman dựa trên tần suất xuất hiện của các ký tự trong dữ liệu, gán mã bit ngắn cho ký tự thường xuyên và mã bit dài hơn cho ký tự ít xuất hiện. Điều này giúp giảm kích thước dữ liệu. Mã hóa LZ77 tìm và loại bỏ sự trùng lặp trong dữ liệu, lưu trữ chỉ vị trí và độ dài của chuỗi trùng lặp cùng với ký tự tiếp theo không trùng lặp. Thuật toán Deflate thực hiện LZ77 trước để tìm và loại bỏ sự

trùng lặp, sau đó áp dụng Huffman để tạo mã bit ngắn hơn. Thông tin về phương pháp nén, bảng mã Huffman, dữ liệu nén và các thông số khác được lưu trữ trong tệp tin nén, giúp tái tạo dữ liệu ban đầu khi giải nén.

2.2.2.1. THUẬT TOÁN LZ77

Thuật toán LZ77 là một thuật toán nén dữ liệu không mất mát dựa trên việc tìm kiếm và loại bỏ sự trùng lặp trong dữ liệu. Ý tưởng chính của LZ77 là sử dụng một "cửa sổ trượt" để tìm các chuỗi giống nhau. Trong quá trình duyệt dữ liệu, thuật toán sử dụng một cửa sổ trượt qua dữ liệu để tìm kiếm các chuỗi trùng lặp. Khi một chuỗi trùng lặp được tìm thấy, thuật toán sẽ lưu trữ một tham chiếu đến chuỗi trùng lặp thay vì lưu trữ toàn bộ chuỗi.

Tham chiếu này thường gồm ba phần: vị trí của chuỗi trùng lặp trong cửa sổ trượt, độ dài của chuỗi, và ký tự đầu tiên sau chuỗi trùng lặp. Điều này giúp giảm kích thước dữ liệu vì tham chiếu thường có kích thước nhỏ hơn so với toàn bộ chuỗi trùng lặp.

Nói cách khác, LZ77 giảm kích thước dữ liệu bằng cách thay thế các chuỗi trùng lặp với các tham chiếu ngắn hơn đến vị trí và độ dài của chúng trong cửa sổ trượt, cùng với ký tự tiếp theo không trùng lặp. Như vậy, LZ77 giúp nén dữ liệu một cách hiệu quả mà không làm mất mát thông tin.

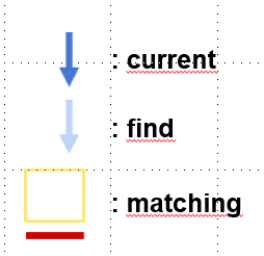
2.2.2.1.1. THUẬT TOÁN

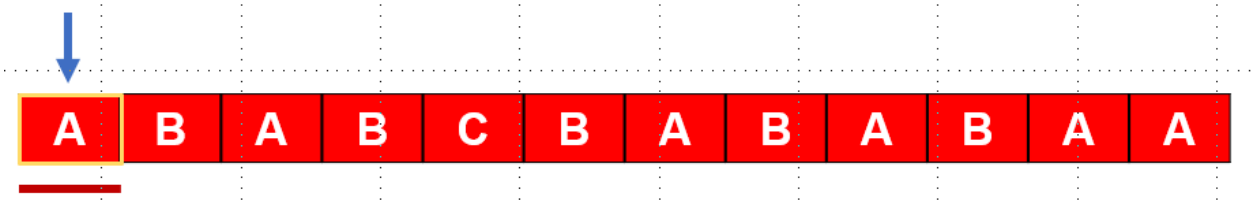
- Bước 1: Duyệt dữ liệu từ trái qua phải.
- Bước 2: Tìm kiếm chuỗi trùng lặp dài nhất xuất hiện trước đó trong từ điển.
- Bước 3: Khi nhìn thấy một chuỗi trùng lặp, thuật toán ghi lại vị trí gần nhất của chuỗi trùng lặp và độ dài của nó.
- Bước 4: Nếu không tìm thấy chuỗi trùng lặp, thuật toán ghi lại dữ liệu gốc tại vị trí hiện tại.
- Bước 5: Lặp lại các bước 2-4 cho khi đã quét qua toàn bộ dữ liệu.

2.2.2.1.2. VÍ DỤ

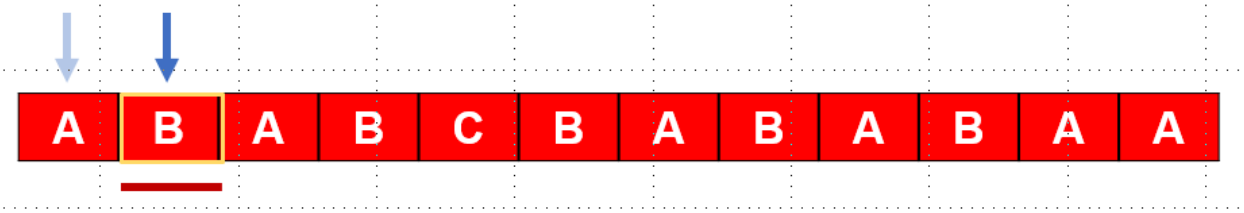
- Để dễ tiếp cận hơn các bước thực hiện thuật toán LZ77, ta sẽ tiến hành nén chuỗi ký tự bằng thuật toán trên : ABABCBABABAA

- Khởi tạo các kí hiệu sau:

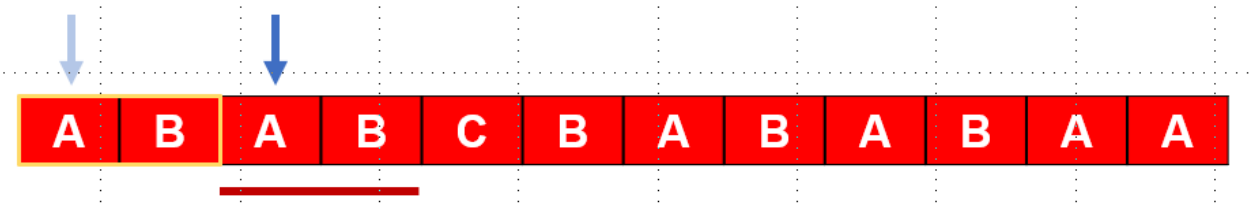




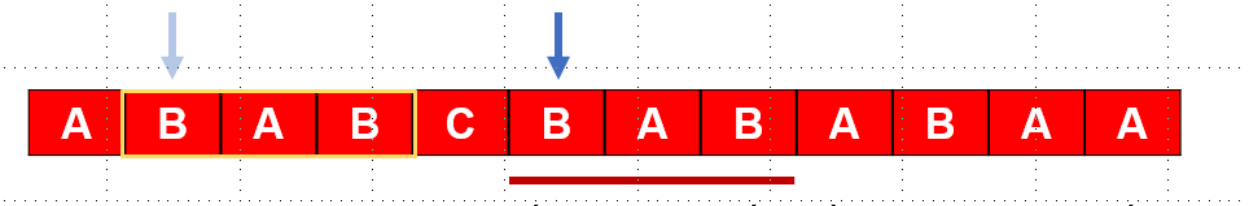
- Bước 1: Bắt đầu duyệt phần tử thứ 1. Tìm kiếm chuỗi trùng lặp dài nhất trước đó, ở đây không có. Ghi lại kí tự tiếp theo không trùng lặp ở đây là A. Ta có LZ77 encode : (0, 0, A)



- Bước 2: Tiếp tục duyệt đến phần tử thứ 2. Tìm kiếm chuỗi trùng lặp dài nhất trước đó, ở đây không có. Ghi lại kí tự tiếp theo không trùng lặp ở đây là B. Ta có LZ77 encode : (0, 0, A) , (0, 0, B)

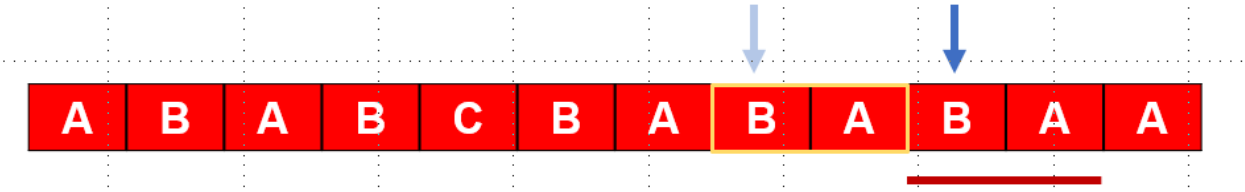


- Bước 3: Tiếp tục duyệt đến phần tử thứ 3. Tìm kiếm chuỗi trùng lặp dài nhất trước đó, ở đây là AB cách vị trí hiện tại là 2 và trùng lặp độ dài là 2. Ghi lại kí tự không trùng lặp tiếp theo là C.
  Ta có LZ77 encode : (0, 0, A) , (0, 0, B), (2, 2, C)



- Bước 4: Vì đã lưu lại kí tự C. Tiếp tục duyệt đến phần tử thứ 6. Tìm kiếm chuỗi trùng lặp dài nhất trước đó, ở đây là BAB cách vị trí hiện tại là 4 và trùng lặp độ dài là 3. Ghi lại kí tự không trùng lặp tiếp theo là A.

  Ta có LZ77 encode : (0, 0, A) , (0, 0, B), (2, 2, C), (4, 3, A)



- Bước 5: Vì đã lưu lại kí tự A. Tiếp tục duyệt đến phần tử thứ 10. Tìm kiếm chuỗi trùng lặp dài nhất trước đó, ở đây là BA cách vị trí hiện tịa là 2 và trụng lặp độ dìa là 2. Ghi lại kí tự không trùng lặp tiếp theo là A.

  Ta có LZ77 encode : (0, 0, A) , (0, 0, B), (2, 2, C), (4, 3, A), (2, 2, A)

- Vì đã encode hết tất cả kí tự nên quá trình nén chuỗi bằng LZ77 kết thúc.

2.2.2.2. THUẬT TOÁN HUFFMAN

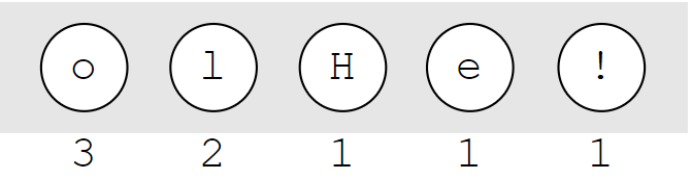
2.2.2.2.1. THUẬT TOÁN

- Bước 1: Đếm tần suất xuất hiện của các phần tử trong chuỗi đầu vào.
- Bước 2: Xây dựng cây Huffman (cây nhị phân mã hóa) với quy ước bên trái mã 0, bên phải mã 1.
- Bước 3: Từ cây Huffman, ta có được các giá trị mã hóa. Lúc này, ta có thể xây dựng chuỗi mã hóa từ các giá trị này.
- Quá trình xây dựng cây Huffman gồm các bước sau:
  - Tạo danh sách chứa các nút lá bao gồm phần tử đầu vào và trọng số nút là tần suất xuất hiện của nó.

- Từ danh sách này, lấy ra 2 phần tử có tần suất xuất hiện ít nhất. Sau đó gắn 2 nút vừa lấy ra vào một nút gốc mới với trọng số là tổng của 2 trọng số ở nút vừa lấy ra để tạo thành một cây.
- Đẩy cây mới vào lại danh sách.
- Lặp lại bước 2 và 3 cho đến khi danh sách chỉ còn 1 nút gốc duy nhất của cây.
- Nút còn lại chính là nút gốc của cây Huffman.

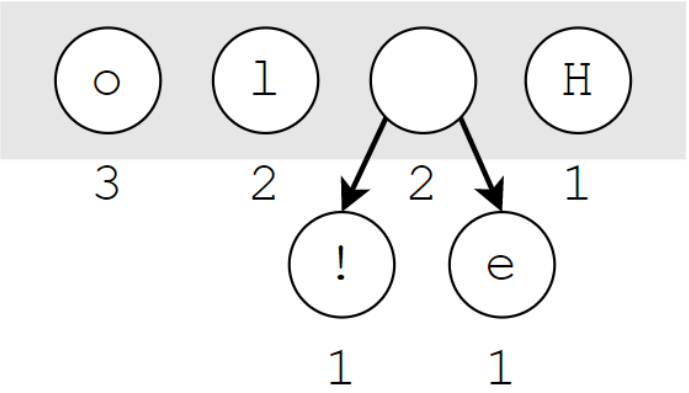
2.2.2.2.2. VÍ DỤ

- Để dễ tiếp cận các bước thực hiện xây dựng cây Huffman, ví dụ cho chữ ***"Hellooo!"***.
  - Bước 2.1: Sau khi đếm tần suất xuất hiện các phần tử đầu vào. Chúng ta tạo danh sách các nút lá với trọng số là tần suất xuất hiện. Danh sách sẽ có 5 phần tử như bên dưới.

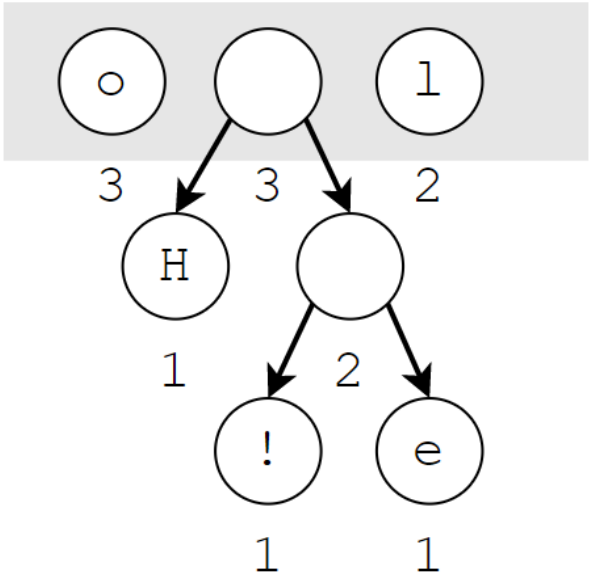


Danh sách ban đầu

  - Bước 2.2 và 2.3: Chọn 2 nút có trọng số thấp nhất, tạo nút gốc mới có trọng số bằng tổng 2 trọng số nút con. Sau đó gắn 2 nút con vào nút gốc và đẩy lại vào danh sách. Danh sách cần được biểu diễn đặc biệt để có thể lấy ra các nút trọng số nhỏ nhất một cách tối ưu nhất.

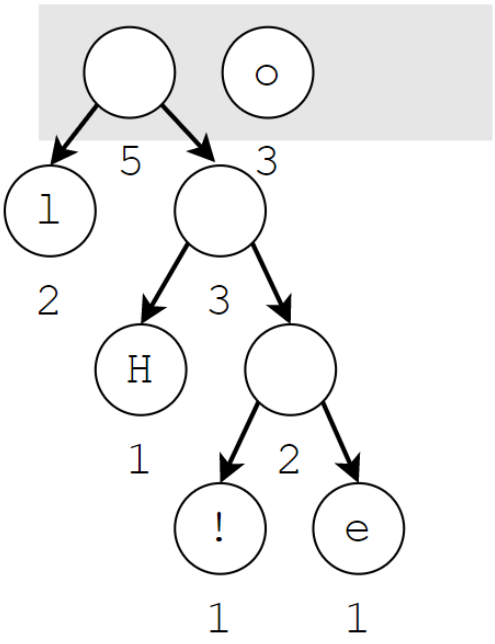


Lần 1



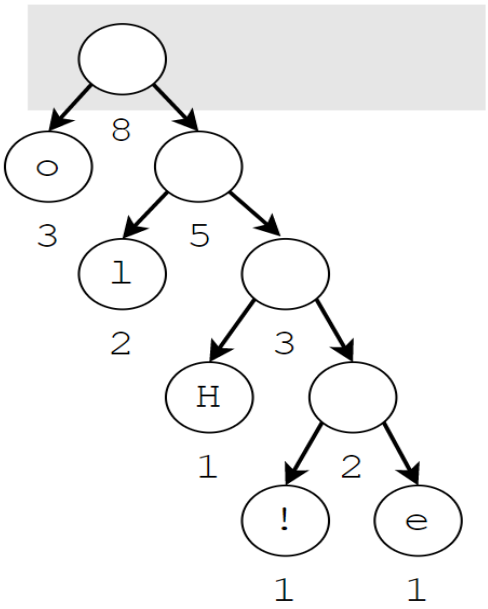
Lần 2

- Bước 2.4: Lặp lại các bước 2.2 và 2.3.



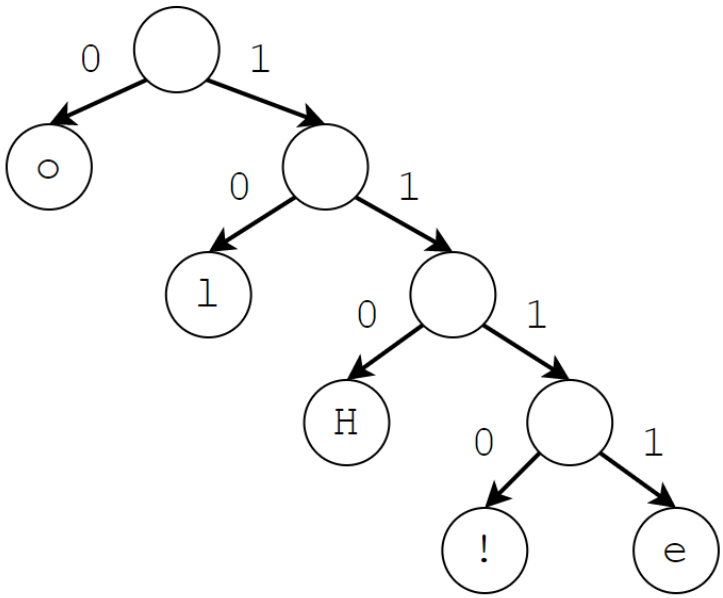
Lần 3

- Bước 2.5: Chỉ còn lại 1 nút trong danh sách, nút này chính là cây Huffman.



Cây còn lại trong danh sách

- Từ cây Huffman, ta có thể suy ra các giá trị mã hóa của từng phần tử bằng cách duyệt cây nhị phân mã hóa.



Cây nhị phân mã hóa

PHẦN 3. THỰC NGHIỆM VÀ ĐÁNH GIÁ

Thử nghiệm trên một tấm ảnh có kích thức 2040x1360 cho ra dung lượng sau khi nén được cải thiện rõ rệt.

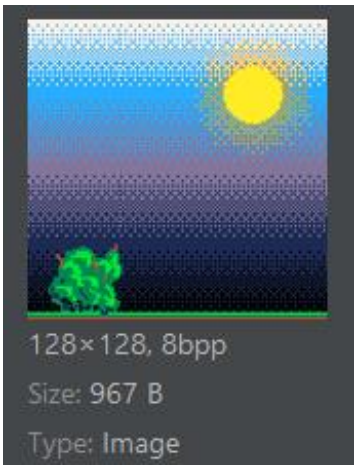


Figure 1 Ảnh trước khi nén

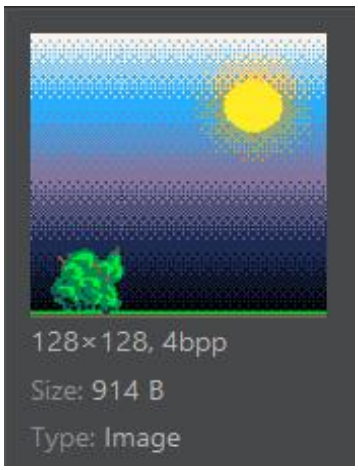


Figure 02 Ảnh sau khi nén

Ngoài ra chúng tôi tiến hành nén và đánh giá trên một số ảnh kích thước lớn hơn. Cụ thể sẽ đánh giá kích thước trước và sau khi nén, tỷ lệ nén và thời gian nén.

Thông tin	Bộ lọc dùng	Kích thước trước khi nén	Kích thước sau khi nén	Tỷ lệ nén	Thời gian nén
beautiful-nature.png	None	4666 KB	4267 KB	1.094	64.047ms
sample.png	Up	3051 KB	2843 KB	1.073	60.438ms
india.png	Sub	2447 KB	2035 KB	1.202	64.399ms
xucxac.png	Sub	222 KB	220 KB	1.009	13.657ms

PHẦN 4. KẾT LUẬN

Không làm mất đi chất lượng hình ảnh. Được phát triển như một định dạng hình ảnh không mất mát, PNG đảm bảo rằng tất cả các dữ liệu hình ảnh ban đầu được lưu giữ, cho phép người dùng trải nghiệm hình ảnh sắc nét và chi tiết.

Kỹ thuật nén PNG chủ yếu sử dụng thuật toán nén LZ77 và mã hóa Huffman để tạo ra tệp PNG nén. Quá trình này giúp loại bỏ các phần trùng lặp trong dữ liệu hình ảnh và sắp xếp lại các thông tin để tạo ra một phiên bản nén hiệu quả hơn.

Tuy nhiên, hiệu suất nén của PNG có thể khá giới hạn đối với ảnh có nhiều màu sắc và chi tiết phức tạp. PNG không thể nén được các loại dữ liệu hình ảnh như ảnh chụp nhanh hoặc hình ảnh đã nén sẵn (như JPEG). Điều này là do PNG tập trung vào việc giữ nguyên chất lượng hình ảnh ban đầu thay vì tối ưu hóa kích thước tệp.

Dù vậy, PNG vẫn được sử dụng rộng rãi trong nhiều trường hợp sử dụng. Đặc biệt là trong việc lưu trữ hình ảnh với độ chính xác cao, chẳng hạn như biểu đồ, biểu đồ dữ liệu, hay biểu tượng đồ họa trên web. PNG cũng hỗ trợ các kênh độ mờ (alpha channel) để tạo ra hình ảnh trong suốt hoặc hiệu ứng đổ bóng.

Tóm lại, nén PNG là một công cụ quan trọng để giảm kích thước tệp ảnh mà không làm mất đi chất lượng hình ảnh ban đầu. Dù hiệu suất nén không cao như một số định dạng hình ảnh khác, nhưng PNG vẫn được sử dụng rộng rãi và hữu ích trong nhiều trường hợp sử dụng.

CHƯƠNG IV. THUẬT TOÁN NÉN ẢNH WEBP

PHẦN 1. GIỚI THIỆU

1.1. ĐẶT VẤN ĐỀ

Trong thời đại số hóa ngày nay, việc tối ưu hóa dữ liệu đặc biệt là hình ảnh trở nên vô cùng quan trọng. Hình ảnh chiếm một phần lớn dữ liệu truyền tải trên mạng, đặc biệt là trên các website. Vì vậy, việc giảm kích thước của hình ảnh mà vẫn giữ được chất lượng là một yếu tố quan trọng đối với hiệu suất website, tốc độ tải trang và trải nghiệm người dùng. Định dạng nén ảnh WebP được phát triển bởi Google, cho phép nén ảnh hiệu quả mà vẫn giữ được chất lượng tốt, thậm chí tốt hơn so với các định dạng hình ảnh khác như JPEG và PNG. Do đó, WebP được chọn làm thuật toán nén ảnh trong phần này.

Nén ảnh bằng WebP không chỉ giúp tối ưu hóa dữ liệu mà còn cung cấp một loạt các lợi ích khác. Khi kích thước của tệp ảnh giảm đi, băng thông cần thiết để tải hình ảnh đó cũng giảm theo, điều này không chỉ giúp tiết kiệm băng thông mà còn tăng tốc độ tải trang, mang lại trải nghiệm tốt hơn cho người dùng. Tốc độ tải trang nhanh hơn không chỉ cung cấp trải nghiệm người dùng suôn sẻ hơn mà còn tăng tỷ lệ chuyển đổi trên website. Bên cạnh đó, việc giảm kích thước hình ảnh cũng giúp tiết kiệm không gian lưu trữ, điều này đặc biệt quan trọng trong thời đại số hóa hiện nay khi dung lượng lưu trữ đóng vai trò quan trọng. Cuối cùng, với khả năng hỗ trợ cả hình ảnh tĩnh và động, WebP không chỉ linh hoạt trong việc tích hợp vào nhiều loại nội dung mà còn giúp nâng cao chất lượng hình ảnh động mà không tăng thêm kích thước tệp.

- Input của bài toán là một hình ảnh trong bất kì định dạng nào (JPEG, PNG, GIF,...).
- Output là hình ảnh sau khi được nén ở định dạng WebP.

1.2. THÁCH THỨC BÀI TOÁN

Mặc dù nén ảnh bằng WebP mang lại nhiều lợi ích rõ ràng, nhưng cũng gặp phải một số thách thức đáng kể. Thách thức lớn nhất có lẽ là vấn đề về khả năng tương thích, với một số trình duyệt cũ không hỗ trợ định dạng này, có thể tạo ra vấn đề về hiển thị hình ảnh cho một số người dùng. Ngoài ra, việc nén hình ảnh bằng WebP đòi hỏi thuật toán phức tạp và tốn kém về mặt tài nguyên máy tính, điều này có thể gây ra khó khăn cho những hệ thống có hạn chế về tài nguyên hoặc những hệ thống yêu cầu thời gian xử lý nhanh. Cuối cùng, mặc dù WebP có thể giữ được chất lượng hình ảnh tốt hơn nhiều so với các định dạng khác khi nén, nhưng việc cân nhắc giữa chất lượng hình ảnh và kích thước tệp vẫn là một thách thức cần được giải quyết trong quá trình nén.

1.3. MỤC TIÊU VÀ PHẠM VI

- Mục tiêu chung ở phần này là xây dựng được các bước của thuật toán nén ảnh webP. Sử dụng các thư viện có sẵn và dữ liệu để tiến hành đánh giá thuật toán nén lossy webP.
- Trong phần này chúng tôi giới hạn phạm vi của đề tài : Dữ liệu chúng tôi sử dụng trong bài toán này là dữ liệu dạng ảnh.

PHẦN 2. THUẬT TOÁN

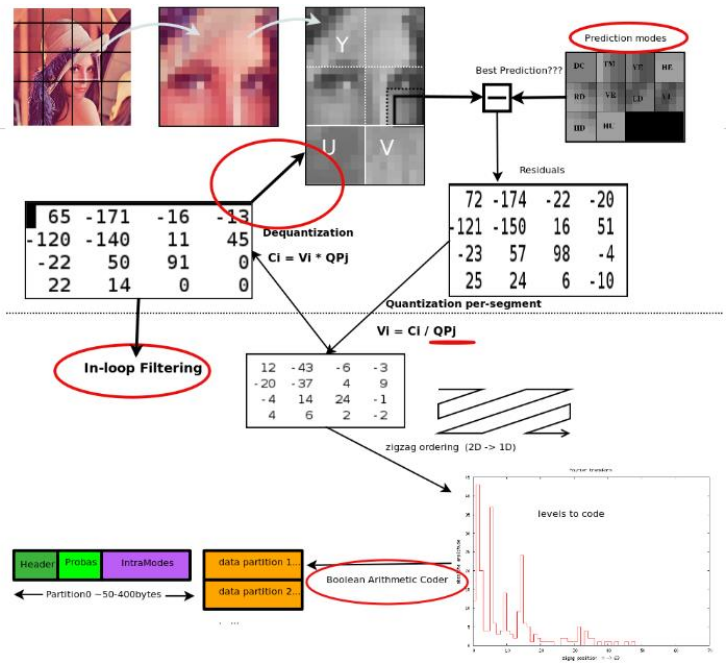
2.1. Ý TƯỞNG

WebP sử dụng hai phương pháp chính để nén hình ảnh: nén mất mác (lossy compression) và nén không mất mác (lossless compression).Trong phương pháp nén mất mác, WebP sử dụng mã hóa dự đoán (predictive coding) để mã hóa hình ảnh. Thuật toán dự đoán các giá trị pixel trong một khối bằng cách sử dụng giá trị của các khối hàng xóm. Thay vì lưu trữ toàn bộ giá trị pixel, chỉ sự khác biệt giữa giá trị dự đoán và giá trị thực tế của từng pixel được mã hóa. Điều này giúp giảm kích thước tệp hình ảnh mà vẫn đảm bảo mức độ chất lượng chấp nhận được. Tuy nhiên, phương pháp này có thể dẫn đến mất mát thông tin nhất định và chất lượng hình ảnh có thể bị ảnh hưởng.

2.2. THUẬT TOÁN

Lossy WebP compression (nén WebP không mất mát) sử dụng mã hóa dự đoán để mã hóa một hình ảnh, cùng phương pháp được sử dụng bởi codec video VP8 để nén các khung chính trong video. Mã hóa dự đoán sử dụng các giá trị trong các khối pixel hàng xóm để dự đoán các giá trị trong một khối và sau đó chỉ mã hóa sự khác biệt giữa dự đoán và giá trị thực tế.

2.2.1. TỔNG QUAN



Thuật toán nén ảnh lossy WebP:

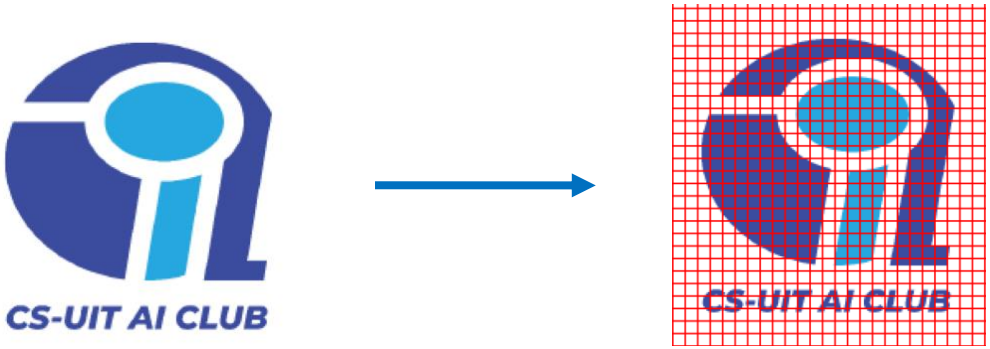
- Bước 1: Block Dividing: Ảnh đầu vào được chia thành nhiều khối macroblock kích thước 4 x 4, 8 x 8 hoặc 16 x 16 pixel và chuyển đổi không gian màu YUV.
- Bước 2: Predictive Coding: Đối với mỗi macroblock, thuật toán sẽ dự đoán giá trị của mỗi pixel dựa trên các pixel xung quanh nó. Sự khác biệt giữa giá trị dự đoán và giá trị thực tế của pixel được gọi là “Residuals”.
- Bước 3: Transform Coding: Các giá trị "residual" sau đó được chuyển đổi từ không gian miền thời gian sang không gian miền tần số thông qua DCT (Discrete Cosine Transform).
- Bước 4: Quantization: Các giá trị sau khi được DCT sẽ được làm tròn (quantization) để giảm độ chính xác, từ đó giảm kích thước dữ liệu cần lưu trữ. Bước này chính là nguồn gốc của việc mất mát thông tin trong nén mất mát.
- Bước 5: Entropy Coding: Một kỹ thuật nén dữ liệu dựa trên sự phân phối xác suất của các mẫu dữ liệu.

2.2.2. CHI TIẾT

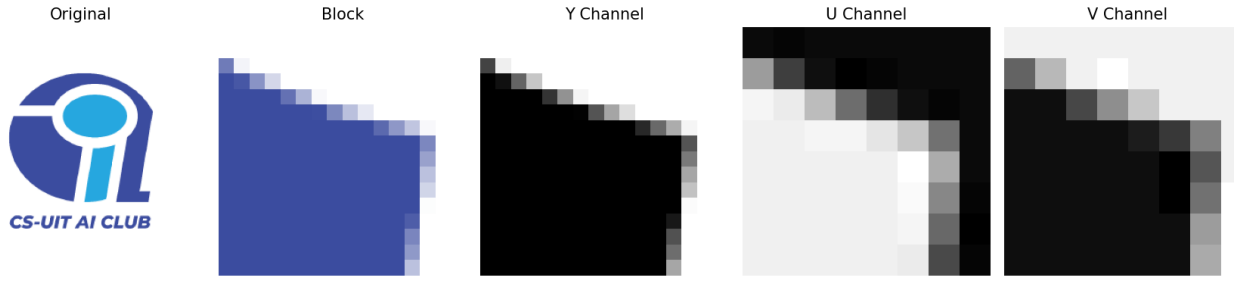
2.2.2.1. BLOCK DIVIDING

Cụ thể, ảnh được chia thành các khối vuông hoặc hình chữ nhật có kích thước như 4x4, 8x8, 16x16 pixel, và các khối này được xử lý độc lập như là các đơn vị nén và xử lý riêng biệt.

Ví dụ:



Sau đó với mỗi macroblock chuyển sang kênh màu YUV để xử lý:



Trong một khối macro thông thường trong webP, có một khối 16x16 pixel cho thành phần luma và hai khối 8x8 pixel cho thành phần chroma. Lý do là vì mắt người có độ nhạy khác nhau đối với các thành phần màu sắc. Thành phần luma (độ sáng) chứa thông tin chi tiết về độ sáng của hình ảnh, trong khi thành phần chroma (độ bão hòa màu) chứa thông tin về sự khác biệt màu sắc.

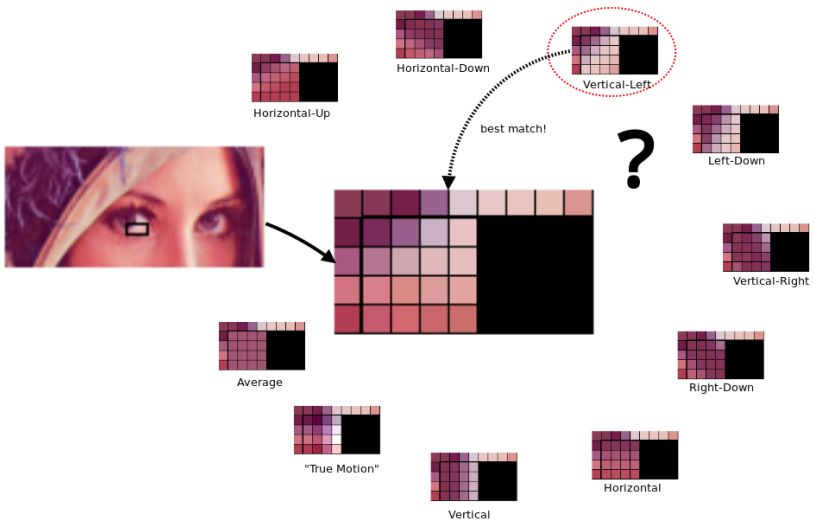
Ví dụ, để tính toán giá trị Y, U và V từ một pixel có giá trị R, G và B sử dụng các công thức chuyển đổi sau:

$$Y = 0.299R + 0.587G + 0.114B$$

$$U = 0.492(B - Y)$$

$$V = 0.877(R - Y)$$

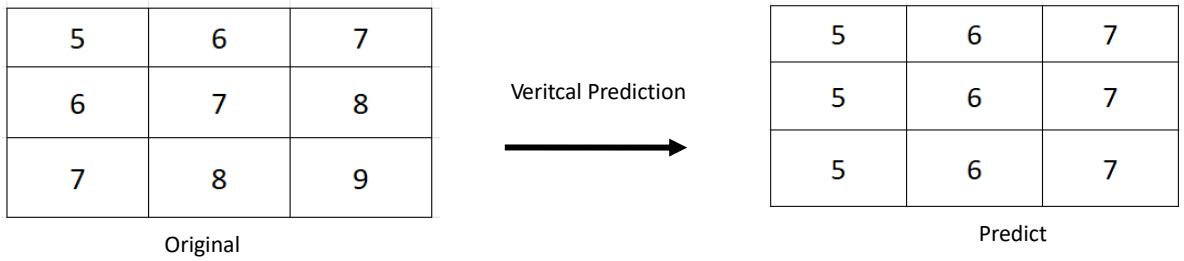
2.2.2.2. PREDICTIVE CODING



Predictive coding là một phương pháp quan trọng trong việc nén hình ảnh định dạng WebP. Kỹ thuật này giảm kích thước tệp hình ảnh bằng cách sử dụng các thuật toán dự đoán giá trị pixel và lưu trữ sự khác biệt giữa giá trị dự đoán và giá trị thực tế.

Quá trình nén bắt đầu bằng việc chia ảnh thành các khối pixel nhỏ. Mỗi khối pixel này được xử lý riêng lẻ. Để dự đoán giá trị của mỗi khối pixel, các thuật toán dự đoán sử dụng thông tin từ các khối pixel xung quanh. Có nhiều phương pháp dự đoán khác nhau có thể được áp dụng, bao gồm lọc thông minh, mô hình dự đoán như hồi quy tuyến tính hoặc học máy.

Sau khi có được dự đoán, sự khác biệt giữa giá trị dự đoán và giá trị thực tế được tính toán. Các khác biệt này, gọi là "hạt nhân" (residual), đại diện cho chênh lệch giữa giá trị dự đoán và thực tế. Từ đó số bit cần phải lưu trữ giảm mạnh. Hạt nhân được lưu trữ và mã hóa một cách hiệu quả.



Ví dụ ta sử dụng Vertical Prediction để dự đoán cho khối 3 x 3 trên. Ta thấy nếu một tấm hình gốc ban đầu cần 4 bit để lưu các giá trị màu. Thì sau khi thực hiện predict thu được ma trận residual thì chỉ cần 2 bit là có thể lưu trữ giá trị màu.

0	0	0
1	1	1
2	2	2

Residual

2.2.2.3. TRANSFORM CODING

Thuật toán Discrete Cosine Transform (DCT) chuyển đổi tín hiệu từ miền không gian sang miền biến đổi, trong đó thông tin tần số của tín hiệu được biểu diễn bằng các hệ số tần số. DCT giúp phân tách thành phần tần số cao và thấp trong tín hiệu. Việc phân tách này cho phép tập trung vào các thành phần tần số quan trọng và có thể loại bỏ hoặc giảm bớt các thành phần không quan trọng.

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N} \quad , \quad 0 \le p \le M-1 \text{ và } 0 \le q \le N-1$$

Với:

$$\alpha_p = \begin{cases} \frac{1}{\sqrt{M}}, & p = 0 \\ \frac{2}{\sqrt{M}}, & 1 \le p \le M-1 \end{cases} \qquad \alpha_q = \begin{cases} \frac{1}{\sqrt{N}}, & q = 0 \\ \frac{2}{\sqrt{N}}, & 1 \le q \le N-1 \end{cases}$$

Trong đó :

- Cho 2 hình ảnh A, B lần lượt là ảnh đầu vào và ảnh đầu ra khi thực hiện DCT
- B_{pq} thể hiện giá trị tại ô (p, q) trên ma trận B tính toán
- A_{mn} thể hiện giá trị giá trị tại ô (m, n) trên ma trận A ban đầu
- M, N kích thước hàng và cột của ma trận A

DCT giúp chuyển đổi dữ liệu hình ảnh từ miền không gian (spatial domain) sang miền tần số (frequency domain), giúp phân loại các thành phần tần số thấp (low-frequency components) và tần số cao (high-frequency components). Các thành phần tần số thấp thường chứa thông tin hình ảnh quan trọng, trong khi các thành phần tần số cao thường chứa chi tiết hình ảnh mà mắt người khó nhìn thấy.

2.2.2.4. QUANTIZATION

Quantization: Sau khi thực hiện DCT, quá trình Quantization sẽ được áp dụng. Đây là bước quan trọng nhất trong việc nén hình ảnh, nơi mà đa số dữ liệu được giảm bớt. Các giá trị DCT sau cùng sẽ được chia cho một ma trận

Quantization (thường được chọn trước) và làm tròn đến số nguyên gần nhất. Quá trình này giúp loại bỏ các thành phần tần số cao mà mắt người khó nhìn thấy, do đó giảm kích thước dữ liệu mà không làm mất đi quá nhiều chất lượng hình ảnh.

Nói cách khác, quantization là quá trình giảm độ chính xác của các giá trị DCT, từ đó giảm kích thước dữ liệu cần lưu trữ. Sự cân nhắc giữa chất lượng hình ảnh và kích thước file là do sự điều chỉnh của ma trận Quantization: nếu quá trình Quantization mạnh hơn (ma trận có giá trị lớn hơn), kích thước file sẽ nhỏ hơn nhưng chất lượng cũng giảm đi và ngược lại.

Công thức tổng quát:

$$F[x, y] = A[x, y] / Q[x, y]$$

Trong đó:

- F[x, y] là giá trị tại vị trí x, y của ma trận sau khi lượng tử
- A[x, y] là giá trị tại vị trí x, y của ma trận trước khi lượng tử
- Q[x, y] là giá trị tại vị trí x, y của ma trận lượng tử

Ví dụ dưới đây là ma trận trước khi lượng tử(trái) và ma trận lượng tử(phải)

150	123
127	135

10	15
20	25

Sau khi thực hiện lượng tử ta thu được ma trận sau khi lượng tử :

15	8
6	5

2.2.2.5. ARITHMETIC CODING

Boolean Arithmetic Coding (BAC), cũng được gọi là entropy coding, là một phần quan trọng khác trong quá trình nén của WebP. Trong ngữ cảnh này, BAC được sử dụng sau khi quá trình DCT và quantization đã được thực hiện.Boolean Arithmetic Coding là một kỹ thuật nén mà ở đó các chuỗi ký tự được thay thế bằng các chuỗi bit ngắn hơn dựa trên tần suất xuất hiện của chúng trong dữ liệu. Kỹ thuật này được gọi là "boolean" vì nó sử dụng logic boolean (AND, OR, NOT) để mã hóa dữ liệu.

Trong WebP, sau quá trình quantization, các giá trị của ma trận đã được mã hóa dựa trên tần suất của chúng trong hình ảnh. Các giá trị phổ biến hơn sẽ được biểu diễn bằng các chuỗi bit ngắn hơn, trong khi các giá trị ít phổ biến hơn sẽ được biểu diễn bằng các chuỗi bit dài hơn. Điều này giúp giảm kích thước tệp hình ảnh.

Thuật toán:

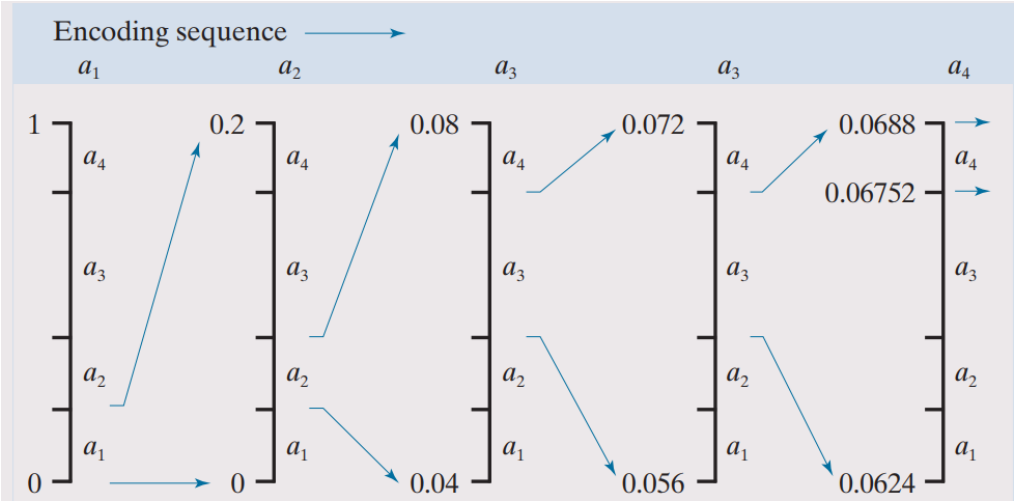
- Khởi tạo: Đặt hai biến low và high đại diện cho khoảng số thực từ 0 đến 1. Điều này tạo ra khoảng cơ bản đại diện cho tất cả chuỗi ký tự có thể xảy ra.
- Tính toán tần suất: Đối với mỗi ký tự trong dữ liệu, tính toán tần suất xuất hiện của nó. Tần suất này thường được biểu diễn dưới dạng một phân số hoặc số thập phân. Kết quả là một biểu đồ tần suất, nơi mỗi ký tự được liên kết với một phạm vi giữa 0 và 1.

  Ví dụ :

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

- Cập nhật khoảng: Đối với mỗi ký tự trong dữ liệu, cập nhật low và high để chúng bao gồm phạm vi của ký tự đó theo biểu đồ tần suất. Cụ thể, low mới sẽ là low cũ cộng với phạm vi của ký tự nhân với sự khác biệt giữa high cũ và low cũ. Tương tự, high mới sẽ là low cũ cộng với phạm vi của ký tự tiếp theo nhân với sự khác biệt giữa high cũ và low cũ.

  Ví dụ:



- Lặp lại: Lặp lại bước 3 cho mỗi ký tự trong dữ liệu. Kết quả cuối cùng là một số trong phạm vi low và high, biểu diễn toàn bộ chuỗi dữ liệu.
- Giải mã: Để giải mã dữ liệu, bạn thực hiện quá trình ngược lại. Bắt đầu với số được mã hóa và biểu đồ tần suất, tìm ký tự mà số đó nằm trong phạm vi của nó, sau đó cập nhật low và high như trên và lặp lại cho đến khi giải mã toàn bộ chuỗi.

❖ Chú ý: Trong thực tế, Arithmetic Coding tối ưu hóa để giảm độ phức tạp và tăng hiệu quả bằng cách sử dụng các biến thể như mô hình ngữ cảnh động cho việc ước lượng tần suất, và dùng số nguyên cố định thay vì số thực để tránh vấn đề độ chính xác và tràn số.

Ví dụ:

- Giả sử chúng ta có một chuỗi ký tự "AABBCD", và chúng ta biết tần suất xuất hiện của mỗi ký tự như sau: A: 0.5, B: 0.25, C: 0.15, D: 0.1.
- Đầu tiên, chúng ta vẽ một đường thẳng từ 0 đến 1 và chia nó thành các phần tương ứng với tần suất của mỗi ký tự.
    - A: 0.0 - 0.5, B: 0.5 - 0.75, C: 0.75 - 0.9, D: 0.9 - 1.0
- Bây giờ, chúng ta bắt đầu mã hóa chuỗi "AABBCD":
    - 'A': Lấy phần từ 0.0 đến 0.5.
    - 'A' tiếp theo: Trong phần từ 0.0 đến 0.5, chúng ta chia phần này lại theo tần suất của mỗi ký tự. Lấy phần từ 0.0 đến 0.25 (đây là nửa đầu của phần từ 0.0 đến 0.5).
    - 'B': Trong phần từ 0.0 đến 0.25, chúng ta chia phần này lại theo tần suất của mỗi ký tự. Lấy phần từ 0.125 đến 0.1875 (đây là nửa sau của phần từ 0.0 đến 0.25).
    - 'B' tiếp theo: Trong phần từ 0.125 đến 0.1875, chúng ta chia phần này lại theo tần suất của mỗi ký tự. Lấy phần từ 0.15625 đến 0.171875 (đây là nửa sau của phần từ 0.125 đến 0.1875).
    - 'C': Trong phần từ 0.15625 đến 0.171875, chúng ta chia phần này lại theo tần suất của mỗi ký tự. Lấy phần từ 0.16375 đến 0.1678125 (đây là 15% của phần từ 0.15625 đến 0.171875).
    - 'D': Trong phần từ 0.16375 đến 0.1678125, chúng ta chia phần này lại theo tần suất của mỗi ký tự. Lấy phần từ 0.16728125 đến 0.1676828125 (đây là 10% của phần từ 0.16375 đến 0.1678125).
    - Chúng ta có thể sử dụng bất kỳ số nào trong phần từ 0.16728125 đến 0.1676828125 để biểu diễn chuỗi "AABBCD". Ví dụ, chúng ta có thể sử dụng số 0.1673.

PHẦN 3. THỰC NGHIỆM VÀ ĐÁNH GIÁ

Thử nghiệm trên một tấm ảnh có kích thức 1920 x 1200 cho ra dung lượng sau khi nén được cải thiện rõ rệt.

Figure Ảnh trước khi nén (4666KB)

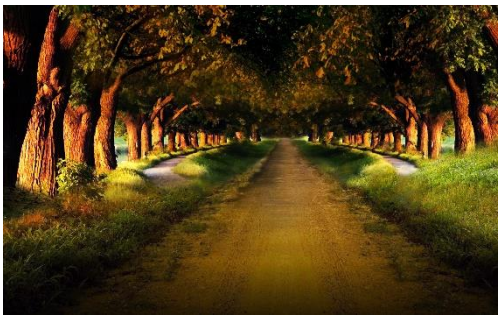
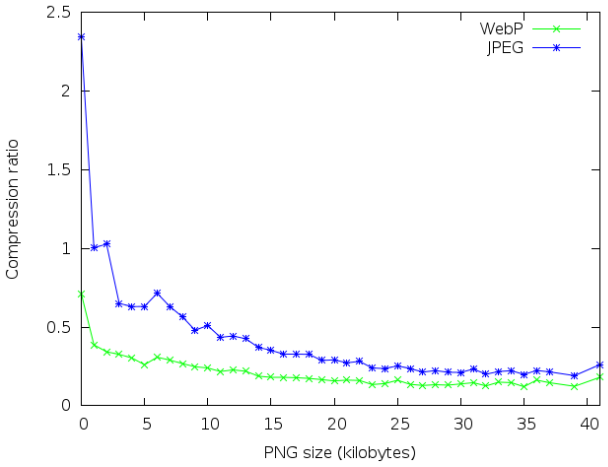
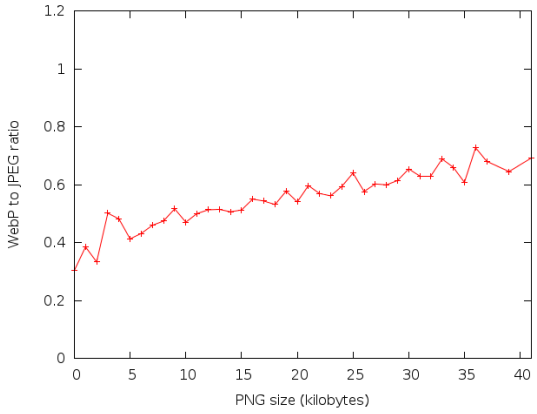


Figure Ảnh sau khi nén (3617KB)

Bảng dưới đây cho thấy tỷ lệ nén của các tệp WebP so với tỷ lệ nén của các tệp JPEG được tạo từ cùng nguồn PNG:





Với bộ dữ liệu : Một bộ sưu tập gồm 1000 biểu tượng dành cho ứng dụng Android đã được sử dụng.

Các biểu tượng được tải xuống dưới dạng các tệp PNG từ trang web cửa hàng ứng dụng Android của Google Play và đại diện cho một ví dụ về các biểu tượng ứng dụng mà chúng ta mong đợi tìm thấy trong việc sử dụng thực tế. Các biểu tượng này có kích thước 124x124 pixel và sử dụng màu RGBA đầy đủ.

PHẦN 4. KẾT LUẬN

Thuật toán nén WebP là một phương pháp hiệu quả để nén hình ảnh trên web. Nó sử dụng một loạt các kỹ thuật nén, bao gồm mã hóa hình ảnh theo dạng không mất mát và mất mát, để giảm kích thước tệp hình ảnh mà vẫn giữ được chất lượng hình ảnh chấp nhận được.

Kết quả của thuật toán nén WebP là tệp hình ảnh có kích thước nhỏ hơn so với định dạng PNG hoặc JPEG thông thường. Điều này có lợi ích rõ ràng cho việc tải trang web nhanh hơn, tiết kiệm băng thông và giảm thời gian tải trang. Ngoài ra, nó cũng giúp giảm lượng lưu trữ hình ảnh trên máy chủ và tiết kiệm không gian đĩa.

Tuy nhiên, việc nén WebP cũng có thể dẫn đến mất mát một số thông tin hình ảnh, đặc biệt là khi sử dụng chế độ nén mất mát cao. Điều này có thể dẫn đến mất mát chi tiết hình ảnh nhỏ, độ phân giải giảm và màu sắc bị biến đổi. Do đó, việc lựa chọn các cài đặt nén phù hợp là quan trọng để đảm bảo cân nhắc giữa kích thước tệp và chất lượng hình ảnh.

Tổng quan, thuật toán nén WebP là một công cụ hữu ích để tối ưu hóa hình ảnh trên web, tăng tốc độ tải trang và giảm tải lưu trữ. Tuy nhiên, cần cân nhắc giữa kích thước tệp và chất lượng hình ảnh để đảm bảo rằng chất lượng hình ảnh không bị ảnh hưởng quá nhiều trong quá trình nén.

BẢNG PHÂN CÔNG CÔNG VIỆC

	Lê Châu Giang	**Ngô Phúc Danh**	**Nguyễn Minh Thư**	**Nguyễn Huỳnh Minh Triết (Nhóm trưởng)**
Thuật toán JPEG	X	X		
Thuật toán JPEG2000	X	X		
Thuật toán PNG			X	X
Thuật toán WEBP			X	X