ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN SỐ 1

COLOR COMPRESSION WITH K-MEANS ALGORITHM

Môn học: **Toán ứng dụng và thống kê cho công nghệ thông tin.**

Mã môn học: **MTH00057.**

Giảng viên hướng dẫn: **Phan Thị Phương Uyên.**

Sinh viên thực hiện: **Nguyễn Đức Vĩnh Hòa.**

Mã số sinh viên: **21127609.**

Lớp: **21CLC05.**

MỤC LỤC:

GIỚI THIỆU VỀ ĐỒ ÁN:

Mục đích:

Nén màu trong hình ảnh là kỹ thuật thu nhỏ kích thước tệp của hình ảnh mà không ảnh hưởng nghiêm trọng đến chất lượng hình ảnh của hình ảnh. Đó là một phương pháp để giảm thiểu không gian lưu trữ và giảm nhu cầu băng thông truyền trong khi lưu trữ, truyền hoặc hiển thị hình ảnh hiệu quả hơn.

Xây dựng chương trình thực hiện việc nén màu sắc của hình ảnh bằng phương pháp K-means clustering.

Chương trình sẽ nén màu sắc của hình ảnh từ một ảnh nhiều màu sắc trở thành một ảnh có số lượng màu thấp mà vẫn giữ được bố cục sợ bộ của ảnh. Người dùng cung cấp các tham số đầu vào là số lượng màu sắc mong muốn, số lần lặp tối đa và loại phân cụm. Kết quả sau khi nén được hiển thị và lưu xuống file theo yêu cầu người dùng.

Ý tưởng xây dựng chương trình:

Ý tưởng hoạt động của chương trình là sử dụng thuật toán K-means clustering để phân cụm màu sắc của hình ảnh ban đầu thành số lượng cụm nhất định. Sau đó, màu sắc của các điểm ảnh được thay thế bằng giá trị trung bình của các centroid của cụm tương ứng. Kết quả là một hình ảnh mới với số lượng màu sắc giảm đi, giúp nén kích thước của hình ảnh và tiết kiệm bộ nhớ.

Mục tiêu của chương trình là giảm số lượng màu sắc trong hình ảnh để nén kích thước và tiết kiệm bộ nhớ.

Phương pháp K-means clustering được áp dụng để phân cụm các điểm ảnh dựa trên màu sắc của chúng.

Các centroids của các cụm sẽ đại diện cho các màu sắc được giữ lại trong hình ảnh sau khi nén.

CHI TIẾT CHƯƠNG TRÌNH:

Các hàm trong chương trình:

Dưới đây là phân tích chi tiết về từng hàm trong chương trình và ý nghĩa của chúng:

1. Hàm ‘read_image(name)’: được sử dụng để đọc hình ảnh từ file đầu vào và chuyển đổi thành đối tượng ‘PIL.Image’ để có thể làm việc với hình ảnh trong các hàm khác của chương trình.

2. Hàm ‘initialize_color(image)’: chuyển đổi hình ảnh từ đối tượng ‘PIL.Image’ sang mảng numpy để thuận tiện cho việc xử lý dữ liệu. Mảng numpy này sẽ có kích thước ‘(m * n, p)’ trong đó ‘m’ và ‘n’ là kích thước của hình ảnh (số điểm ảnh), và ‘p’ là số kênh màu.

3. Hàm ‘initialize_centroids(colorPoint, k_clusters, clusterT)’: Khởi tạo ra các centroids ban đầu cho thuật toán K-means. Các centroids sẽ đại diện cho các màu sắc được giữ lại trong quá trình nén màu sắc. Phương pháp khởi tạo centroids có thể là ngẫu nhiên hoặc dựa trên các điểm ảnh.

Nếu loại phân cụm là 'random', hàm sẽ tạo ngẫu nhiên các centroids trong khoảng giá trị màu sắc.

Nếu loại phân cụm là 'in_pixels', hàm sẽ chọn ngẫu nhiên các điểm ảnh từ mảng màu sắc làm centroids ban đầu.

4. Hàm ‘EuclideanDistance(x, y)’: Tính khoảng cách Euclidean giữa hai điểm ảnh dựa trên thông tin màu sắc của chúng. Khoảng cách Euclidean được sử dụng để đo lường sự tương đồng giữa các điểm ảnh trong thuật toán K-means.

5. Hàm ‘UpdateLabel(img_1d, centroids)’: Cập nhật nhãn cho từng điểm ảnh dựa trên khoảng cách Euclidean từ các centroids. Nhãn này xác định cụm mà điểm ảnh thuộc về trong quá trình K-means clustering.

6. Hàm ‘UpdateCentroids(img_1d, centroids, labels)’: Cập nhật các centroids dựa trên nhãn của từng điểm ảnh. Các centroids được cập nhật bằng cách tính giá trị trung bình của các điểm ảnh trong cùng một cụm.

7. Hàm ‘kmeans(img_1d, k_clusters, max_iter, init_centroids)’: Thực hiện triển khai thuật toán K-means để thực hiện quá trình nén màu sắc. Nó chứa các bước lặp để cập nhật nhãn và centroids cho đến khi hội tụ hoặc đạt đến số lần lặp tối đa. Cuối cùng, nó trả về các centroids cuối cùng và nhãn tương ứng.

8. Hàm ‘if __name__ == "__main__":’ kiểm tra xem chương trình có được chạy như là chương trình chính hay không. Nếu đúng, nó yêu cầu người dùng cung cấp các thông số đầu vào, thực hiện quá trình nén màu sắc và lưu hình ảnh sau khi nén vào file.

Một số đặc điểm của chương trình:

- Chương trình cho phép người dùng tùy chỉnh số lượng màu sắc mong muốn, số lần lặp tối đa và loại phân cụm.

- Hình ảnh sau khi nén được hiển thị và lưu xuống file theo yêu cầu người dùng.

- Chương trình không xử lý trường hợp khi không tìm thấy file hình ảnh đầu vào hoặc không thể lưu kết quả nén xuống file. Cần thêm xử lý lỗi phù hợp.

Một số cải tiến có thể thêm vào chương trình:

- Mở rộng hỗ trợ định dạng hình ảnh: Thêm khả năng hỗ trợ nén và lưu các định dạng hình ảnh khác.

- Đánh giá độ chính xác và hiệu suất: Thêm tính năng để đánh giá độ chính xác và hiệu suất của quá trình nén, bằng cách tính toán độ tương đồng sau nén và đo thời gian thực thi.

- Tối ưu hóa thuật toán.

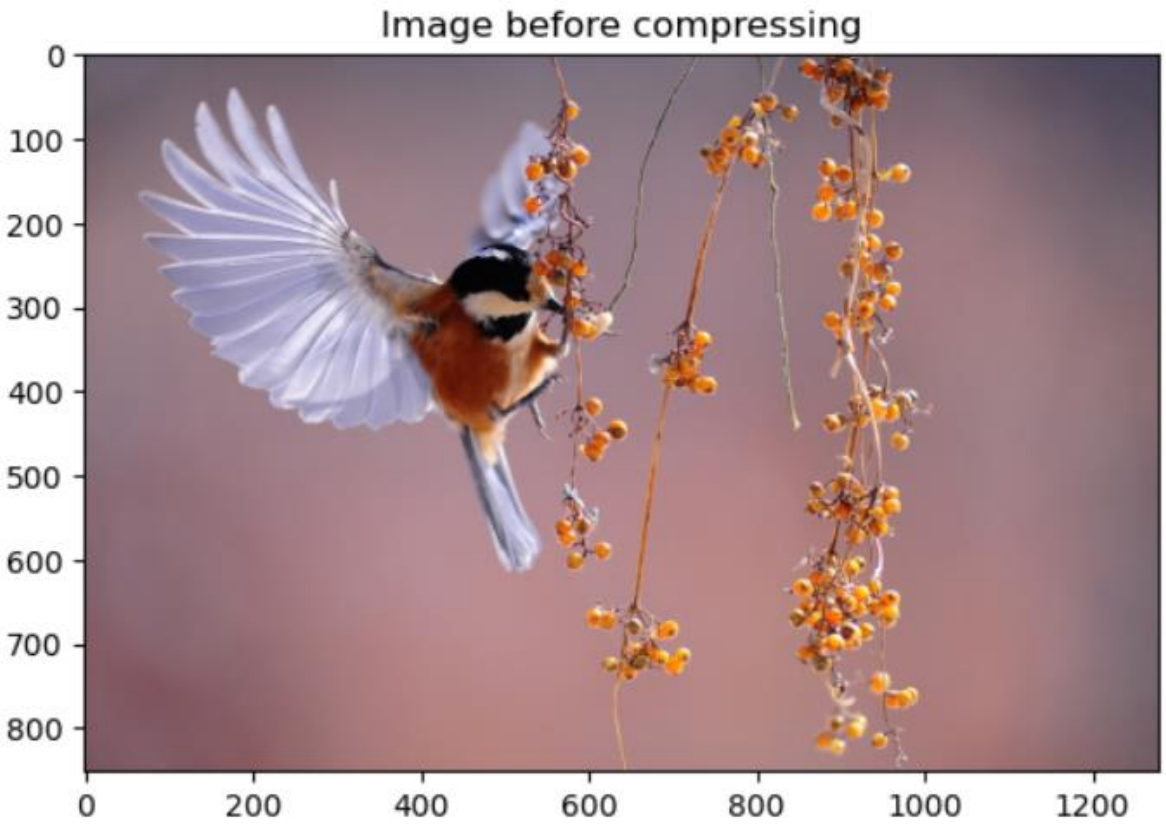
CHẠY CHƯƠNG TRÌNH, THỬ NGHIỆM VÀ KẾT QUẢ

Phân cụm random và ảnh đơn giản:

Thử nghiệm chương trình với các thông số sau đây:

- Số lượng màu k= 3, 5, 7.
- Số vòng lặp tối đa được đặt là 1000.
- Loại phân cụm sẽ là random.
- Ảnh đơn giản, số lượng màu ít, mật độ phân bố màu không dày đặc.

Ảnh ban đầu:



Với k=3:

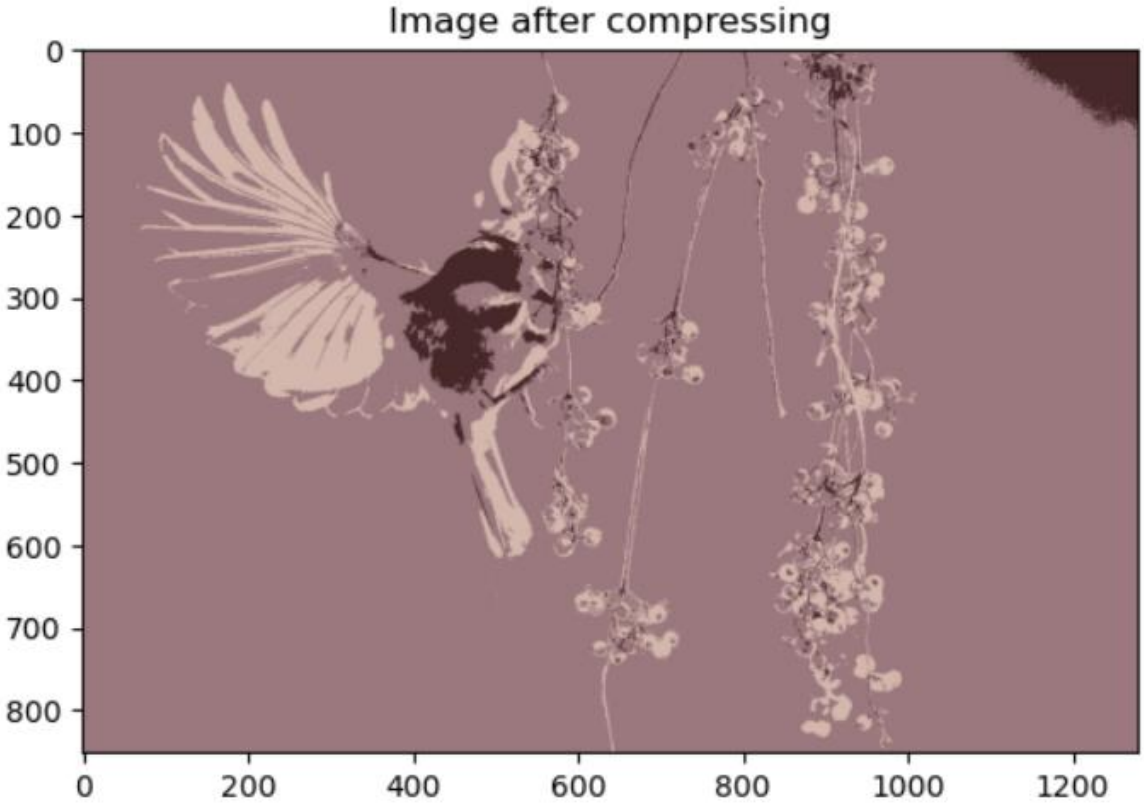


Image after compressing
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000
1200

Với k=5:

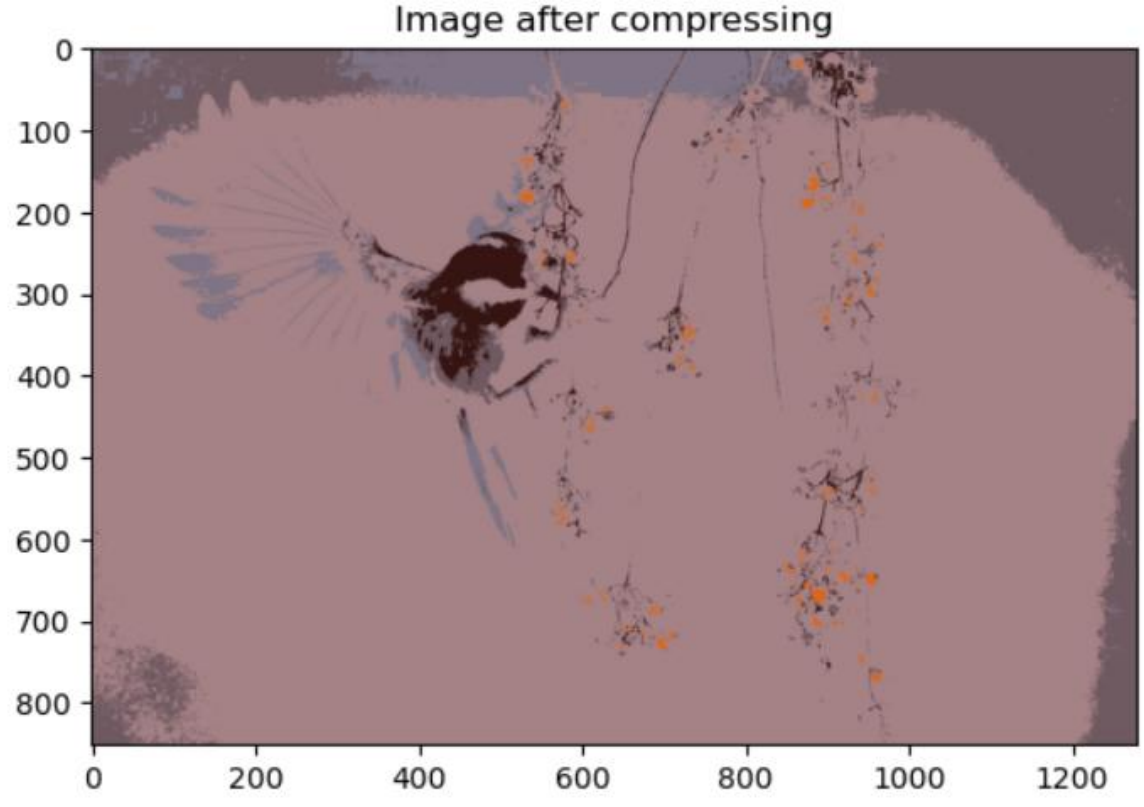
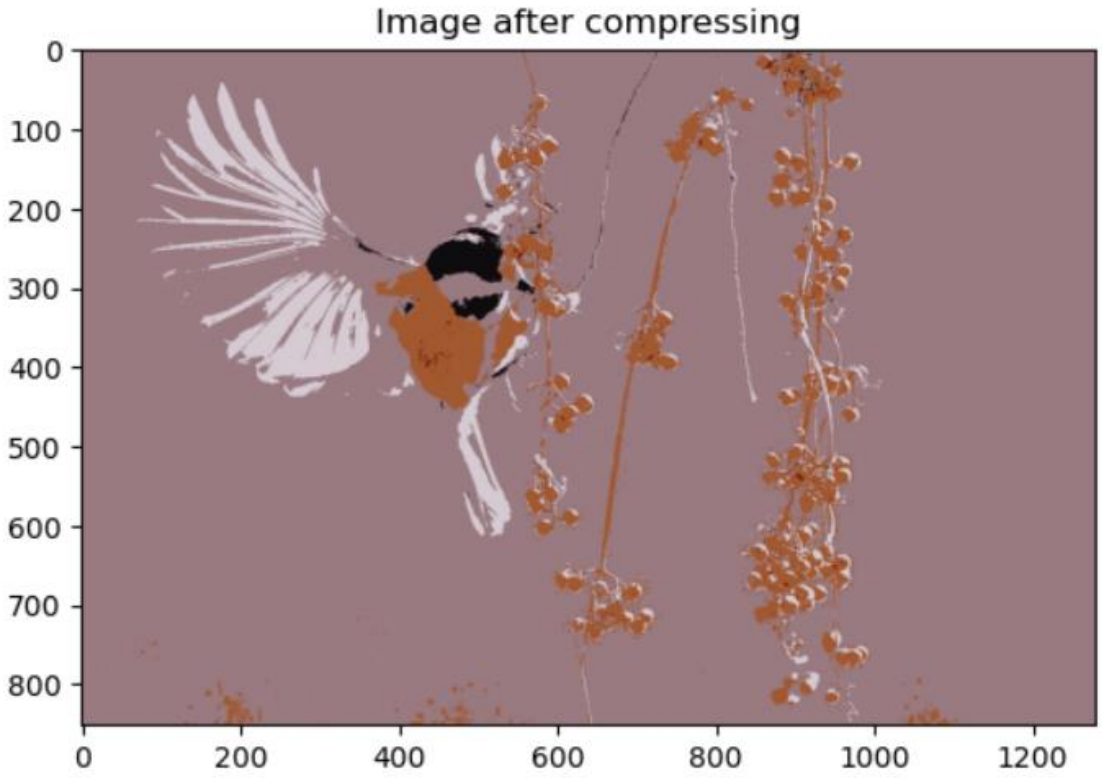


Image after compressing
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000
1200

Với k=7:



Nhận xét:

→ Có thể thấy được sự khác biệt dần từ mức k=3 đến k=7.

→ Với số lượng màu cụm ít hơn, hình ảnh mới có xu hướng tập trung vào các màu sắc chủ đạo, chỉ sử dụng một số lượng nhỏ các màu sắc đại diện.

→ Ngược lại, với số lượng màu cụm nhiều hơn, hình ảnh mới có độ phong phú màu sắc cao hơn và sự đa dạng màu sắc rõ rệt hơn.

Phân cụm in_pixels và ảnh đơn giản:

Thử nghiệm chương trình với các thông số sau đây:

- Số lượng màu k= 3, 5, 7.
- Số vòng lặp tối đa được đặt là 1000.
- Loại phân cụm sẽ là in_pixels.
- Ảnh đơn giản, số lượng màu ít, mật độ phân bố màu không dày đặc.

Với k=3:

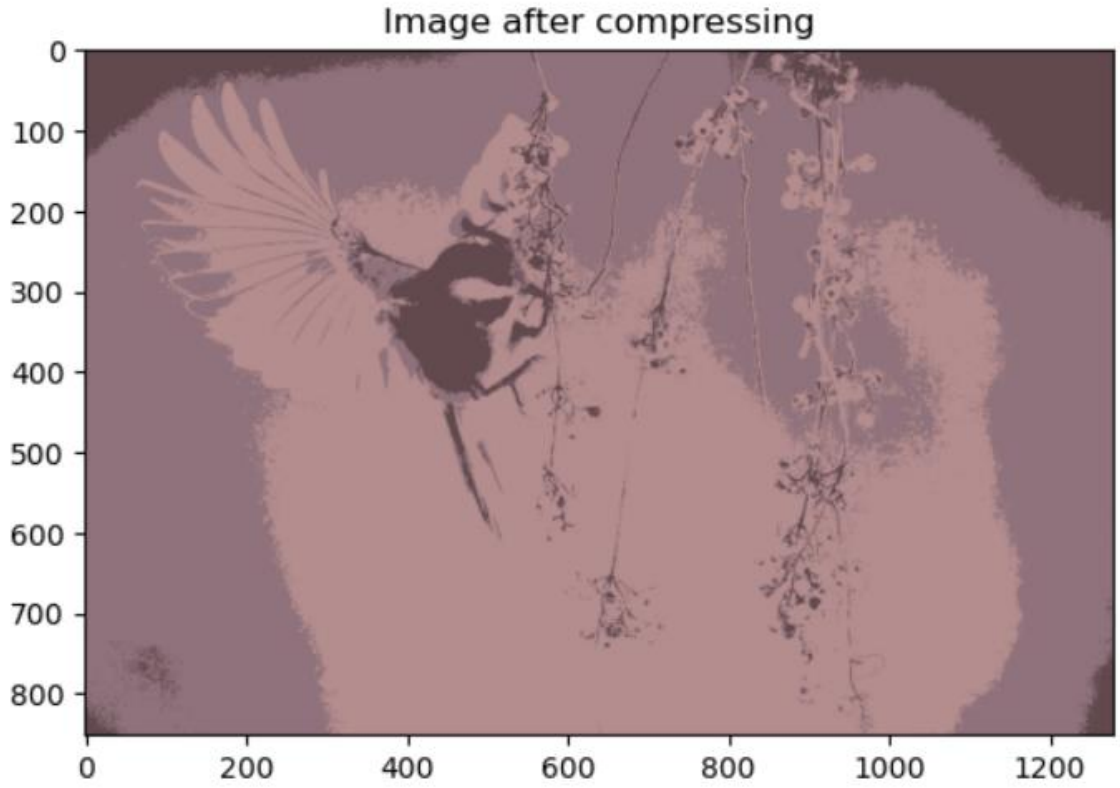


Image after compressing
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000
1200

Với k=5:

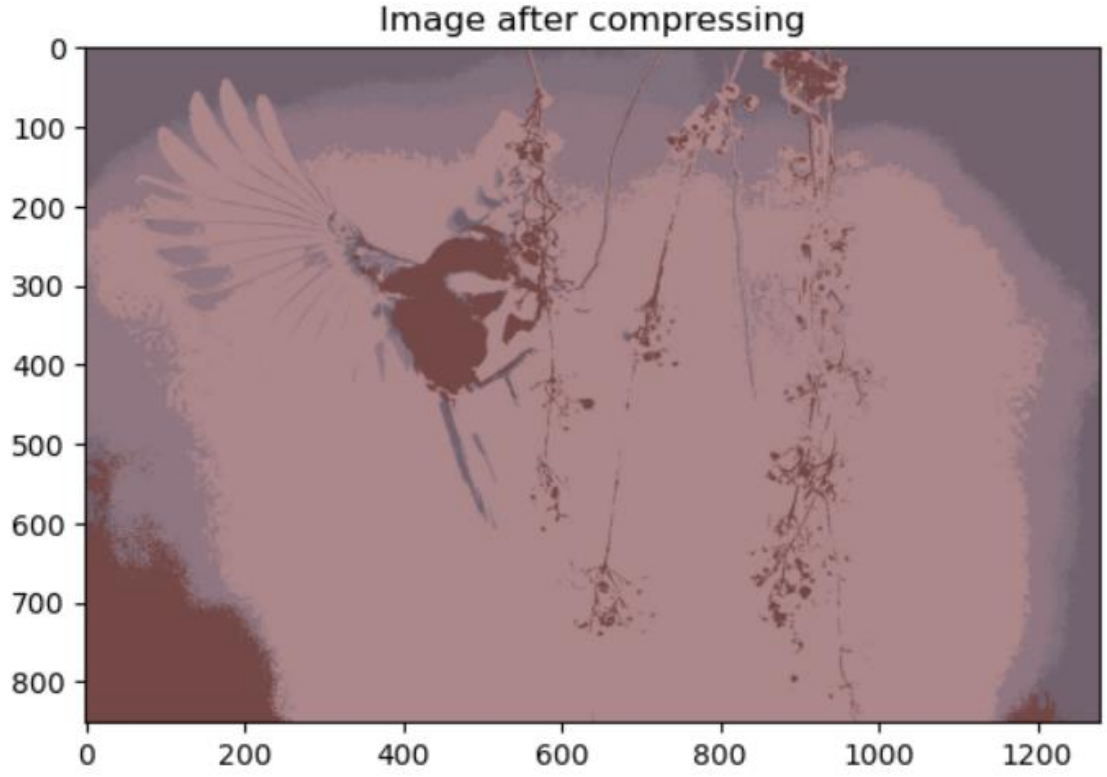
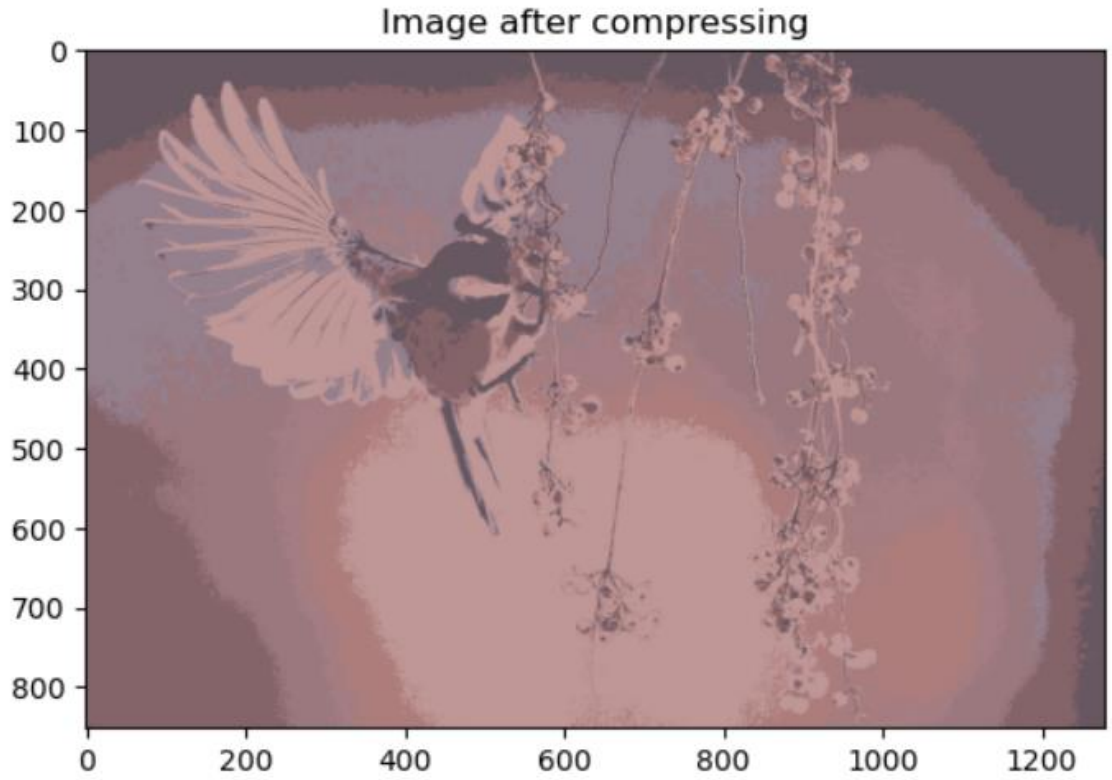


Image after compressing
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000
1200

Với k=7:



Nhận xét:

→ Các chi tiết của bức ảnh được thể hiện rõ hơn đối với khi sử dụng là random.

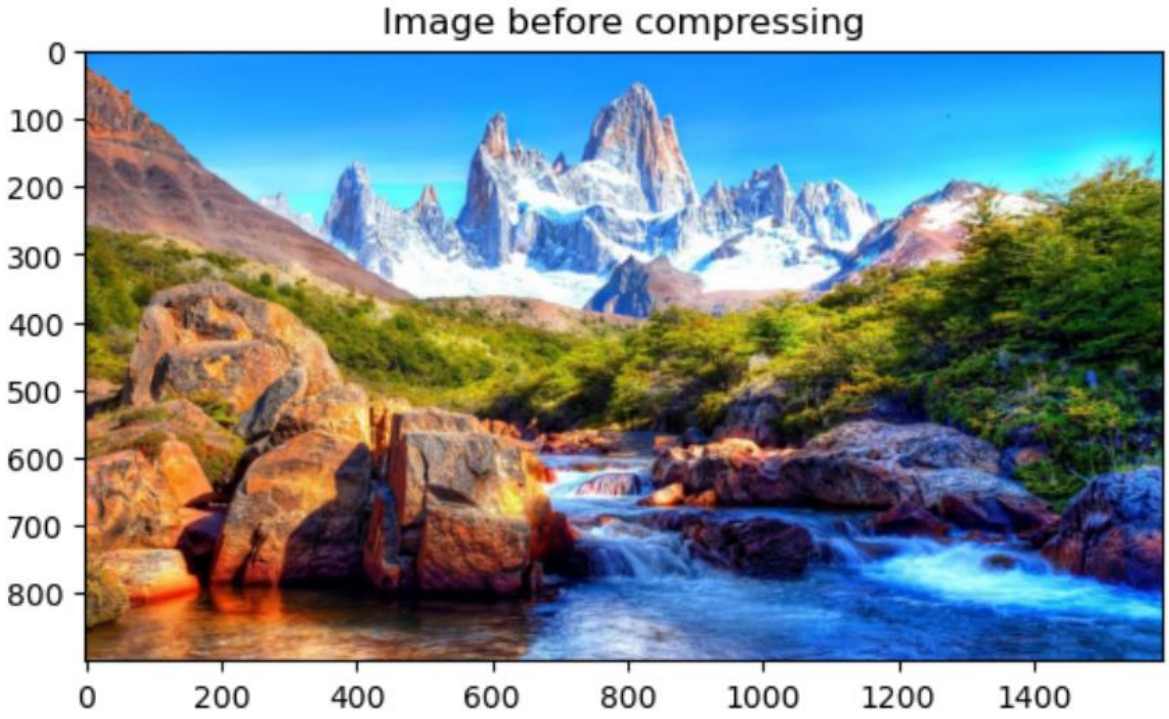
→ Chất lượng ảnh cũng được cải thiện hơn so với random.

Phân cụm random và ảnh phức tạp:

Thử nghiệm chương trình với các thông số sau đây:

- Số lượng màu k= 3, 5, 7.
- Số vòng lặp tối đa được đặt là 1000.
- Loại phân cụm sẽ là random.
- Ảnh phức tạp, số lượng màu lớn, mật độ phân bố màu dày đặc.

Ảnh ban đầu:



Với k=3:

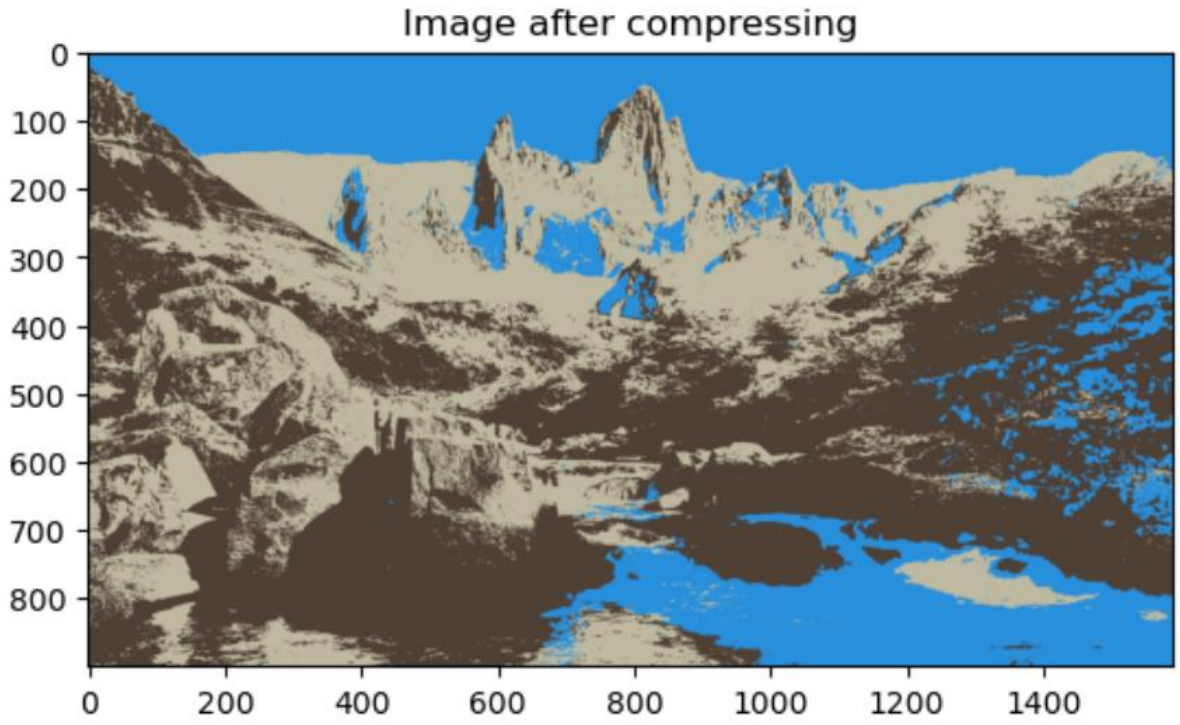


Image after compressing
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000
1200
1400

Với k=5:

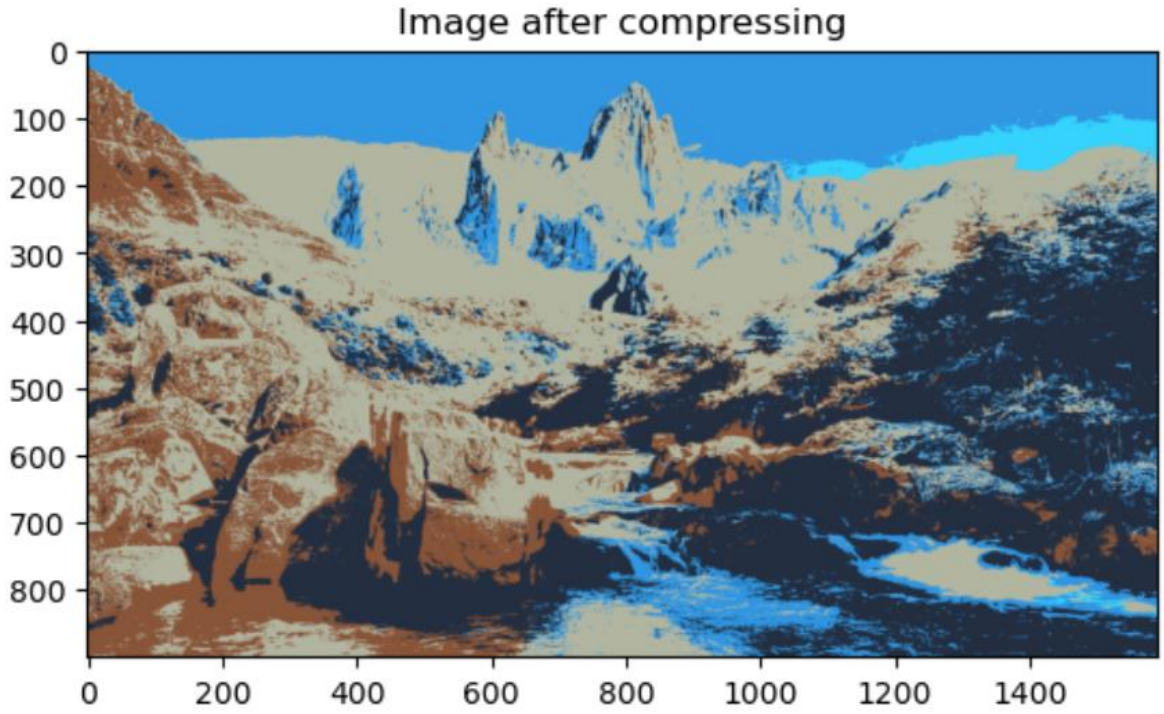
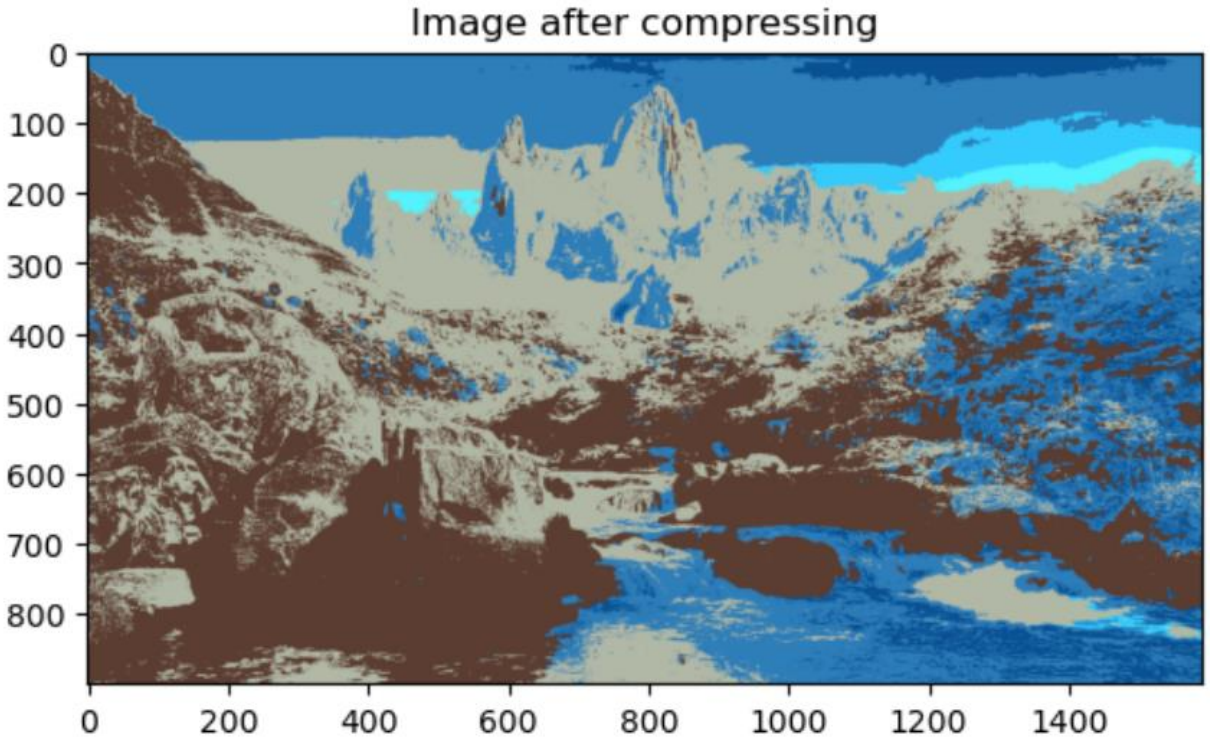


Image after compressing
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000
1200
1400

Với k=7:



Nhận xét:

→ Có thể thấy được sự khác biệt dần từ mức k=3 đến k=7.
→ Với số lượng màu cụm ít hơn, hình ảnh mới có xu hướng tập trung vào các màu sắc chủ đạo, chỉ sử dụng một số lượng nhỏ các màu sắc đại diện.
→ Ngược lại, với số lượng màu cụm nhiều hơn, hình ảnh mới có độ phong phú màu sắc cao hơn và sự đa dạng màu sắc rõ rệt hơn.
→ Nhưng do số lượng màu của ảnh nhiều nên phần lớn màu sắc của bức ảnh không được thể hiện chính xác.

Phân cụm in_pixels ảnh phức tạp:

Thử nghiệm chương trình với các thông số sau đây:

- Số lượng màu k= 3, 5, 7.
- Số vòng lặp tối đa được đặt là 1000.
- Loại phân cụm sẽ là random.
- Ảnh phức tạp, số lượng màu lớn, mật độ phân bố màu dày đặc.

Với k=3:

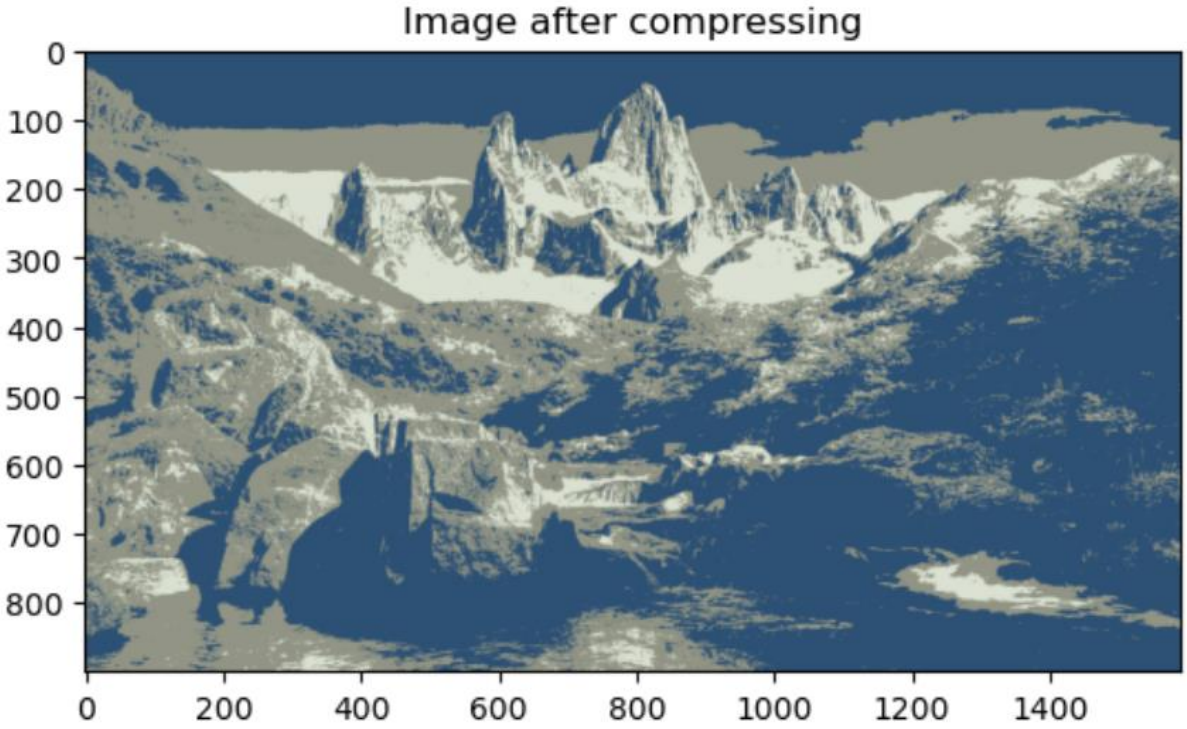


Image after compressing
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000
1200
1400

Với k=5:

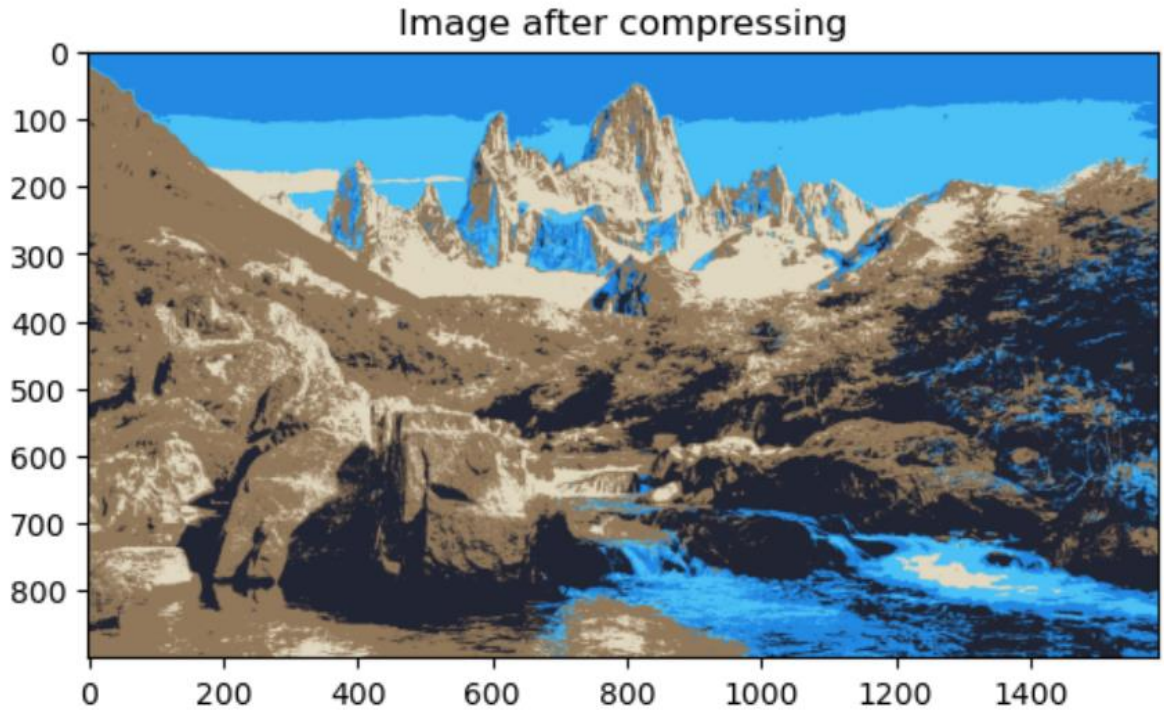
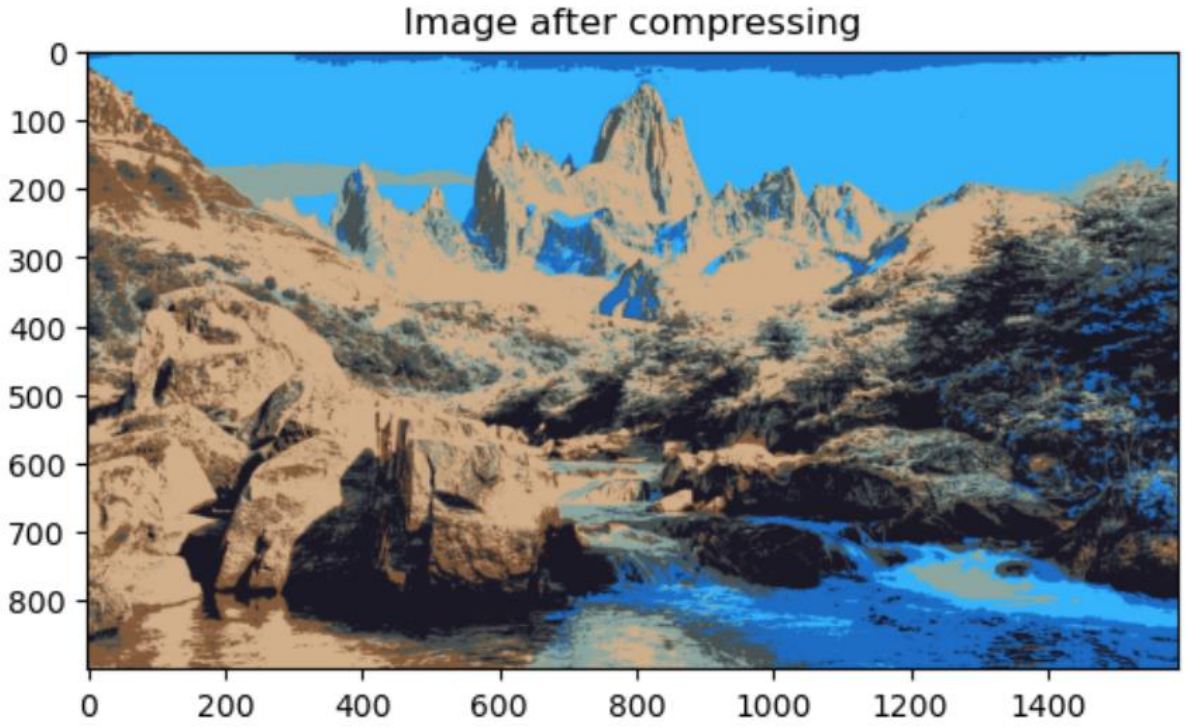


Image after compressing
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000
1200
1400

Với k=7:



Nhận xét:

→ Các chi tiết của bức ảnh được thể hiện rõ hơn đối với khi sử dụng là random.
→ Chất lượng ảnh cũng được cải thiện hơn so với random.
→ Hình ảnh trông hoàn chỉnh hơn so với random.
→ Màu được phối hợp lí hơn random.

Nguyên nhân có sự khác biệt giữa random và in_pixels:

- Trường hợp của random, nghĩa là các trung tâm cụm (centroids) được chọn ngẫu nhiên từ miền giá trị màu sắc của hình ảnh ban đầu. Sự ngẫu nhiên trong việc chọn điểm trung tâm ban đầu dẫn đến sự phân tán ngẫu nhiên của các cụm màu ban đầu.
- Trường hợp của in_pixels, nghĩa là các trung tâm cụm (centroids) được chọn ngẫu nhiên trong các pixel của ảnh. Việc chọn điểm trung tâm ban đầu từ các điểm ảnh trong hình ảnh ban đầu tạo ra sự phụ thuộc và tính phức tạp hơn trong quá trình tính toán. Điều này có thể dẫn đến việc thuật

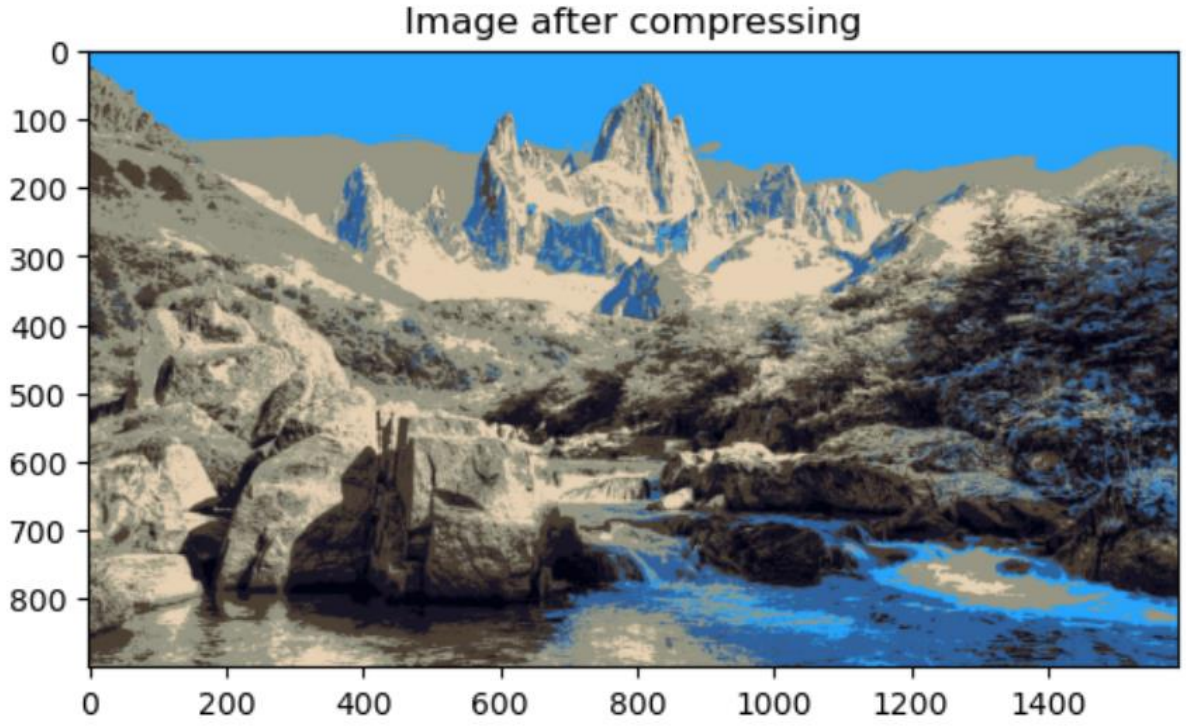
toán cần nhiều lần lặp hơn để hội tụ và đạt được kết quả tối ưu.

Số lần lặp tối đa có ảnh hưởng nhiều đến kết quả không?

Thử nghiệm chương trình với các thông số sau đây:

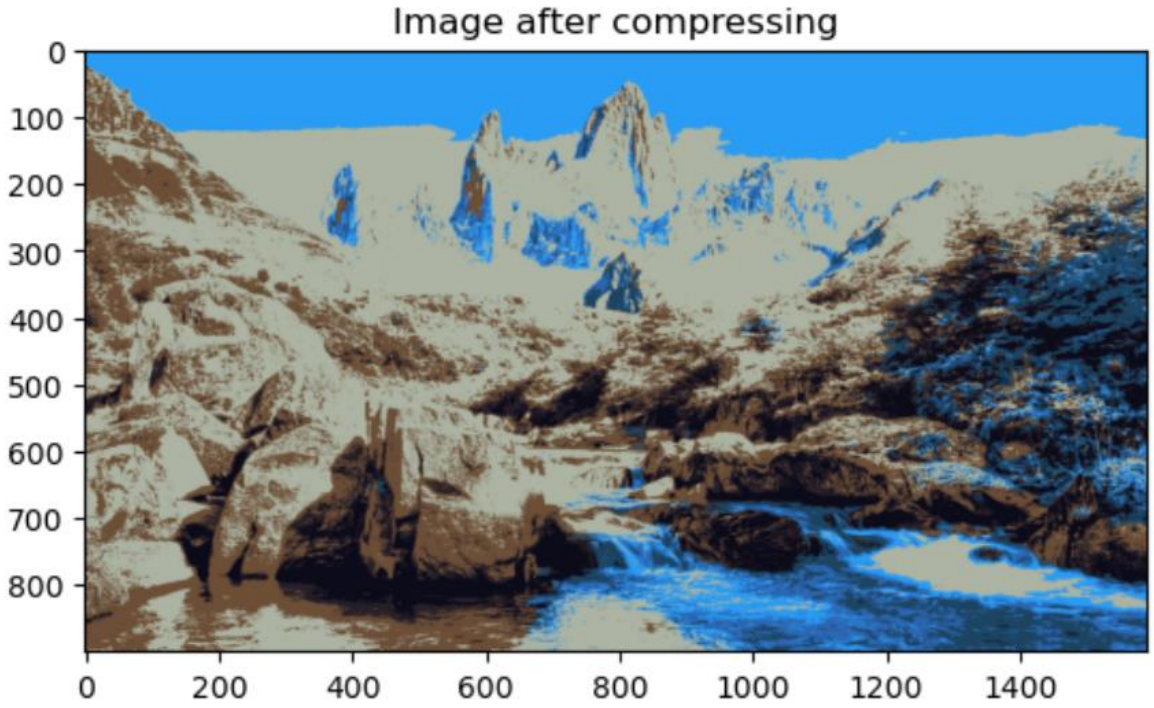
- Số lượng màu k= 7.
- Số vòng lặp tối đa được đặt là 1000 và 100000.
- Loại phân cụm sẽ là in_pixels.
- Ảnh phức tạp, số lượng màu lớn, mật độ phân bố màu dày đặc.

Với maxIter=1000:



Thời gian hoàn thành: 22.43

Với maxIter=100000:



Thời gian hoàn thành: 22.70

Nhận xét:

- Về độ chi tiết 2 hình ngang nhau
- Phân bổ màu và mật độ màu tương đương nhau.
- Có lẽ do kích thước ảnh chưa đủ lớn và số lượng màu cho ra còn ít nên số vòng lặp tối đa không ảnh hưởng nhiều đến chất lượng ảnh. Nhưng qua tìm hiểu vẫn có thể kết luận được rằng:
    - Số vòng lặp tối đa quyết định số lần lặp tối đa mà thuật toán K-means được thực hiện.
    - Khi maxIter càng lớn, thuật toán có cơ hội lặp lại quá trình cập nhật nhãn và centroids nhiều lần hơn để hội tụ tốt hơn.
    - Nếu thuật toán chưa hội tụ sau khi đã thực hiện số lần lặp tối đa, kết quả sẽ là trạng thái của centroids gần đến việc hội tụ, nhưng không phải là hội tụ hoàn toàn.
    - Điều này có thể dẫn đến sự mất mát chi tiết màu sắc và kết quả nén không được tối ưu.
    - Một max_iter nhỏ có thể dẫn đến sự hội tụ không đủ và kết quả không tối ưu.

Bảng so sánh thời gian chạy chương trình ở các test case:

	Ảnh đơn giản			Ảnh phức tạp		
Số lượng màu	K=3	K=5	K=7	K=3	K=5	K=7
Random	7.80s	12.73s	19.34s	11.22s	19.85s	25.86s
In_pixels	7.87s	13.50s	17.69s	10.41s	16.39s	22.72s

KẾT LUẬN:

- Chương trình thực hiện quá trình nén màu sắc của hình ảnh, giúp giảm kích thước tệp hình ảnh và tiết kiệm bộ nhớ lưu trữ.
- Phương pháp sử dụng K-means clustering là một phương pháp đơn giản và hiệu quả để thực hiện nén màu sắc trong hình ảnh.
- Ứng với mỗi giá trị 'k', thuật toán K-means sẽ phân đoạn hình ảnh thành 'k' cụm khác nhau dựa trên màu sắc của các điểm ảnh và chọn ngẫu nhiên 'k' điểm ảnh trong hình ảnh làm các điểm trung tâm ban đầu.
- Ảnh kết quả sẽ hiển thị sự phân đoạn với k màu sắc khác nhau tương ứng với k cụm được tìm thấy.
- Các điểm ảnh thuộc cùng một cụm sẽ có màu sắc tương tự nhau và tạo ra các vùng có màu sắc giống nhau trên hình ảnh kết quả.
- Quá trình hội tụ của thuật toán K-means phụ thuộc vào điều kiện dừng và cách cập nhật các điểm trung tâm. Khi số lần lặp tới giới hạn hoặc khi khoảng cách giữa các điểm trung tâm không thay đổi đáng kể, thuật toán được coi là đã hội tụ.

TÀI LIỆU THAM KHẢO:

https://machinelearningcoban.com/2017/01/01/kmeans/

https://en.wikipedia.org/wiki/K-means_clustering

https://doi.org/10.1016/j.patrec.2004.04.007

https://machinelearningcoban.com/2017/01/01/kmeans/

https://www.geeksforgeeks.org/image-compression-using-k-means-clustering/

https://youtu.be/RaTve-ddaps