

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO TIẾN ĐỘ ĐỒ ÁN
Toán ứng dụng và thống kê cho công nghệ thông tin
21CLC04

Đồ án

COLOR COMPRESSION



Giáo viên hướng dẫn

Nguyễn Văn Quang Huy
Ngô Đình Hy
Nguyễn Đình Thúc

Thành viên

Lý Nhật Hào - 21127041

NIÊN KHOÁ 2022 - 2023

LỜI CẢM ƠN

Để hoàn thành được bài báo cáo này, chúng em đã nhận được sự giúp đỡ rất nhiều từ phía thầy cô giảng viên, trợ giảng và bạn bè. Nay em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến giảng viên môn Toán ứng dụng và thống kê cho Công Nghệ Thông Tin lớp 21CLC4, Khoa Công nghệ thông tin :

- Giảng viên **Nguyễn Đình Thúc**
- Giảng viên **Ngô Đình Huy**
- Giảng viên **Nguyễn Văn Quang Huy**

Các thầy đã đồng hành, đã luôn quan tâm, hướng dẫn và truyền đạt, cung cấp kiến thức, tài liệu và các thủ thuật cần thiết để em có thể hoàn thành đồ án !

Trong quá trình thực hiện đồ án không thể tránh khỏi những thiếu sót. Chúng em rất mong nhận được nhiều ý kiến đóng góp từ các giảng viên và bạn bè để đồ án ngày càng hoàn thiện hơn!



THÀNH VIÊN NHÓM BÁO CÁO

X

Lý Nhật Hào

Xin chân thành cảm ơn!



MỤC LỤC

LỜI CẢM ƠN	1
Chương I: Giới Thiệu Đề Án & Thuật Toán K-MEANS	4
1.1. Thông tin cá nhân sinh viên thực hiện đồ án:	4
1.2. Tổng quát yêu cầu đồ án:	4
1.3. Đôi nét về Thuật Toán K-MEANS:	4
1.4. Các Bước Thực Hiện Của Thuật Toán K-means:	5
CHƯƠNG II: ÁP DỤNG THUẬT TOÁN K-MEANS VÀO BÀI TOÁN COLOR COMPRESSION	6
2.1. BÀI TOÁN COLOR COMPRESSION:	6
2.2. Nhận Xét Về Thuật Toán K-MEANS Trong Việc Giải Bài Toán Color Compression	6
2.2.1. Ưu điểm	6
2.2.2. Khuyết điểm	7
CHƯƠNG III: CÀI ĐẶT THUẬT TOÁN VÀ HƯỚNG DẪN SỬ DỤNG CHƯƠNG TRÌNH	8
3.1. Pseudocode:	8
3.2. Hướng Dẫn Sử Dụng Chương Trình Color Compression:	11
3.3. Nhận xét kết quả thuật toán:	14
• ĐỐI VỚI ẢNH CÓ ĐỘ PHÂN GIẢI THẤP 953x960 (ẢNH ĐẦU):	14
• ĐỐI VỚI ẢNH CÓ ĐỘ PHÂN GIẢI CAO 2119 x 1414 (ẢNH THỨ 2)	15
CHƯƠNG IV: MỘT SỐ THUẬT TOÁN KHÁC ỨNG DỤNG TRONG COLOR COMPRESSION	16
4.1. Giới thiệu chung	16
4.2. Principal Component Analysis (PCA)	17
4.2.1. Thuật Toán:	17
4.2.2. Ưu điểm và Khuyết điểm của thuật toán PCA:	18
4.3. Thuật Toán Tối Ưu Nhất Cho Bài Toán Color Compression	19
CHƯƠNG V: TÀI LIỆU THAM KHẢO	20

Chương I: Giới Thiệu Đồ Án & Thuật Toán K-MEANS

1.1. Thông tin cá nhân sinh viên thực hiện đồ án:

Họ & Tên	Lý Nhật Hào
MSSV	21127041
EMAIL	lnhao21@clc.fitus.edu.vn

1.2. Tổng quát yêu cầu đồ án:

Đồ án cuối kỳ phải đáp ứng ít nhất các yêu cầu sau:

- Trong đồ án này, bạn được yêu cầu cài đặt chương trình giảm số lượng màu cho ảnh sử dụng thuật toán K-MEANS.
- Các thư viện được phép sử dụng là:
 - ‘NumPy’ (tính toán ma trận).
 - ‘PIL’ (đọc, ghi ảnh).
 - ‘matplotlib’ (hiển thị ảnh).
- Sinh viên cần viết chương trình **Main** cho phép:
 - Người dùng nhập vào tên tập tin ảnh mỗi lần chương trình thực thi (gợi ý sử dụng ‘input()’ trong Python)
 - Lựa chọn định dạng lưu ảnh đầu ra gồm: ‘png’, ‘pdf’

1.3. Đôi nét về Thuật Toán K-MEANS:

Thuật toán K-Means là một thuật toán phân cụm dữ liệu trong học không giám sát (clustering), được sử dụng để phân nhóm các điểm dữ liệu vào các nhóm khác nhau dựa trên các đặc trưng của chúng. Thuật toán K-Means được các nhà khoa học máy tính gọi là thuật toán “centroid-based” vì nó sử dụng centroid của các nhóm để xác định những điểm dữ liệu nào thuộc về cùng một nhóm.

Thuật toán K-Means hoạt động bằng cách đưa ra một số lượng k nhóm ban đầu (k được cho trước), sau đó phân bố các điểm dữ liệu vào các nhóm khác nhau dựa trên khoảng cách của chúng đến centroid của từng nhóm. Sau đó, centroid của mỗi nhóm được cập nhật dựa trên các điểm dữ liệu thuộc về nhóm đó, và quá trình này lặp lại đến khi không có sự thay đổi nào trong các centroid của các nhóm.

1.4. Các Bước Thực Hiện Của Thuật Toán K-means:

1. **Khởi tạo ngẫu nhiên k trung tâm cụm ban đầu:** Thuật toán K-means bắt đầu với việc khởi tạo ngẫu nhiên k trung tâm cụm ban đầu. K là một số nguyên dương được xác định trước đó và được coi là số lượng cụm.
2. **Gán từng điểm dữ liệu vào cụm gần nhất:** Sau khi khởi tạo các trung tâm cụm ban đầu, thuật toán K-means sẽ gán từng điểm dữ liệu vào cụm gần nhất (có khoảng cách nhỏ nhất) với trung tâm cụm tương ứng.
3. **Cập nhật lại trung tâm cụm cho mỗi cụm:** Sau khi gán từng điểm dữ liệu vào cụm gần nhất, thuật toán K-means sẽ cập nhật lại trung tâm cụm cho mỗi cụm bằng cách tính trung bình của tất cả các điểm dữ liệu trong cụm đó.
4. **Lặp lại các bước gán và cập nhật cho đến khi không có sự thay đổi nào trong trung tâm cụm hoặc đạt đến số lần lặp tối đa được thiết lập trước:** Sau khi cập nhật lại trung tâm cụm cho mỗi cụm, thuật toán K-means sẽ tiếp tục gán từng điểm dữ liệu vào cụm gần nhất và cập nhật lại trung tâm cụm cho mỗi cụm. Quá trình này được lặp lại cho đến khi không có sự thay đổi nào trong trung tâm cụm hoặc đạt đến số lần lặp tối đa được thiết lập trước.
5. **Sau khi hoàn tất quá trình lặp, thuật toán K-means trả về k trung tâm cụm tối ưu và phân chia các điểm dữ liệu vào k cụm.**

CHƯƠNG II: ÁP DỤNG THUẬT TOÁN K-MEANS VÀO BÀI TOÁN COLOR COMPRESSION

2.1. BÀI TOÁN COLOR COMPRESSION:

- **Bài toán color compression có thể được giải quyết bằng thuật toán K-Means.** Bài toán này đặt ra vấn đề làm thế nào để giảm kích thước của một hình ảnh mà không làm giảm chất lượng của nó quá nhiều.
- Thuật toán **K-Means** có thể được sử dụng để giảm số lượng màu sắc trong hình ảnh. Cụ thể, ta có thể sử dụng thuật toán K-Means để phân nhóm các pixel trong hình ảnh thành các cluster (nhóm) dựa trên giá trị màu sắc của chúng. Sau đó, mỗi cluster sẽ được đại diện bằng một màu sắc duy nhất, được tính bằng trung bình của tất cả các pixel trong cluster đó. Khi đó, ta chỉ cần sử dụng các màu sắc này để tạo ra hình ảnh mới, thay vì sử dụng tất cả các màu sắc ban đầu.
- Việc giảm số lượng màu sắc sẽ giảm kích thước của hình ảnh và tạo ra một hình ảnh mới có chất lượng tương đối giống với hình ảnh ban đầu. Tuy nhiên, việc chọn số lượng cluster (K) phù hợp là rất quan trọng để đảm bảo rằng màu sắc của hình ảnh mới vẫn giữ được tính đa dạng và độ chính xác đáng chấp nhận.

2.2. Nhận Xét Về Thuật Toán K-MEANS Trong Việc Giải Bài Toán Color Compression

2.2.1. Ưu điểm

- **Nhanh và Đạt Hiệu Quả Cao:**

Thuật toán K-means rất nhanh và hiệu quả kể cả đó là đối với dữ liệu lớn bởi vì thuật toán trên có khả năng xử lý các bức ảnh có độ phân giải lớn mà không yêu cầu nhiều tài nguyên tính toán hoặc bộ nhớ khổng lồ.

- **Có Khả Năng Nén Ảnh Đáng Chú Ý:**

Thuật toán K-means thể hiện sự nén ảnh hiệu quả thông qua việc làm giảm số lượng màu sắc có trong một bức ảnh xuống rất thấp mà vẫn mà không làm giảm quá nhiều chất lượng của hình ảnh. Từ đó, bức ảnh sẽ có độ nén cao.

- **Có Khả Năng Tùy Chỉnh Số Lượng Màu Sắc:**

K-means cho phép tùy chỉnh số lượng màu sắc khác nhau cho từng phần của hình ảnh, do đó giúp tạo ra những hình ảnh nén với chất lượng tốt hơn.

- **Dễ Hiểu Và Dễ Triển Khai:**

Thuật toán K-means là một thuật toán đơn giản và dễ hiểu, do đó nó dễ triển khai trên các nền tảng khác nhau.

2.2.2. Khuyết điểm

- **Không Xác Định Được Giá Trị Tối Ưu Toàn Cục Cho Thuật Toán:**

K-means có thể rơi vào điểm cực tiểu địa phương, tức là nó có thể không tìm được kết quả tối ưu toàn cục sau một lần chạy. Do đó, cần chạy nhiều lần và chọn kết quả tốt nhất để cải thiện kết quả của thuật toán.

- **Khó Khăn Trong Việc Chọn Số Lượng Màu Sắc Phù Hợp:**

Số lượng màu sắc quá ít có thể làm mất đi nhiều chi tiết quan trọng trong hình ảnh, trong khi số lượng màu sắc quá nhiều sẽ không giảm được kích thước của hình ảnh dẫn đến thuật toán không hiệu quả.

- **Vị Trí Của Tâm Cụm Lê Thuộc Vào Vị Trí Khởi Tạo:**

K-means yêu cầu xác định số lượng cụm trước đó, và việc chọn sai số lượng cụm có thể dẫn đến kết quả phân cụm không tốt.

- **Gặp Vấn Đề Với Ảnh Nhiều Hoặc Có Quá Nhiều Chi Tiết Nhỏ:**

K-means nhạy cảm với các điểm dữ liệu nhiều và các giá trị ngoại lai.

CHƯƠNG III: CÀI ĐẶT THUẬT TOÁN VÀ HƯỚNG DẪN SỬ DỤNG CHƯƠNG TRÌNH

3.1. Pseudocode:

1. Class Kmeans để thực hiện giải thuật K-means clustering
2. Trong class Kmeans:
 - a. Phương thức `init()` dùng để khởi tạo số lượng cụm màu, số lần lặp tối đa và số lần lặp ngẫu nhiên
 - b. Phương thức `cen_ran()` dùng để tạo ngẫu nhiên các centroid ban đầu
 - c. Phương thức `nearing_k()` dùng để tính toán khoảng cách giữa các điểm và các centroid
 - d. Phương thức `wcentr_mean()` dùng để tính toán lại vị trí của các centroid
 - e. `warning_compute()` dùng để tính toán tổng khoảng cách giữa các điểm và các centroid
 - f. Phương thức `solving()` dùng để thực hiện giải thuật K-means clustering
 - g. Phương thức `predict()` dùng để dự đoán nhãn của các điểm

```
class Kmeans:  
    def __init__(self, kcluster_n, i_max=999, i_rand=999):  
        self.kcluster_n = kcluster_n  
        self.i_max = i_max  
        self.i_rand = i_rand  
    def cen_ran(self, X):  
        np.random.RandomState(self.i_rand)  
        rand_idx = np.random.permutation(X.shape[0])  
        centr = X[rand_idx[:self.kcluster_n]]  
        return centr  
    def nearing_k(self, X, centr):  
        distance_k = np.zeros((X.shape[0], self.kcluster_n))  
        for index in range(self.kcluster_n):  
            distance_k[:,index] = np.square(np.linalg.norm(X - centr[index, :], axis=1))  
        return np.argmin(distance_k, axis=1)  
    def centr_mean(self, X, idx):  
        centr = np.zeros((self.kcluster_n, X.shape[1]))  
        for index in range(self.kcluster_n):  
            centr[index, :] = np.mean(X[idx == index, :], axis=0)  
        return centr  
    def warning_compute(self, X, idx, centr):  
        distance = np.zeros(X.shape[0])  
        for index in range(self.kcluster_n):  
            distance[index == idx] = np.linalg.norm(X[index == idx] - centr[index], axis=1)  
        return np.sum(np.square(distance))  
    def solving(self, X):  
        self.centr = self.cen_ran(X)  
        for i in range(self.i_rand):  
            previous_centroids = self.centr  
            self.idx = self.nearing_k(X, previous_centroids)  
            self.centr = self.centr_mean(X, self.idx)  
            if np.all(self.centr == previous_centroids):  
                break  
        self.error = self.warning_compute(X, self.idx, self.centr)
```

3. Hàm **Formating()** để cho phép người dùng lựa chọn định dạng file xuất ra (PNG, JPG hoặc PDF)

```
def Formating():
    print('***** CHOICE OF FILE OUT FORMAT *****')
    print('Enter 1 : Output as JPG file')
    print('Enter 2 : Output as PNG file')
    print('Enter 3 : Output as PDF file')
    print('*****')
    decision = int(input('Enter your choice: '))
    if (decision == 1):
        return '.jpg'
    if (decision == 2):
        return '.png'
    if (decision == 3):
        return '.pdf'
```

4. Các hàm **kcluster3()**, **kcluster5()**, **kcluster7()**, và **kcluster9()** dùng để thực hiện nén ảnh với số cụm màu tương ứng là 3, 6, 7 và 9 màu. Trong các hàm kể trên, phương thức hoạt động là:

- Mở ảnh và chuyển đổi thành một **mảng numpy**.
- Thực hiện giải thuật K-means clustering với số lượng cụm màu là 3, 5, 7 hoặc 9 trên mảng numpy của ảnh.
- Chuyển đổi lại mảng numpy của ảnh đã nén về định dạng **uint8** và **reshape** lại thành kích thước ban đầu.
- Hiển thị ảnh gốc và ảnh đã nén lên cùng một hình với 2 cột bằng hàm **subplot** của **matplotlib**.
- Xuất ảnh đã nén ra file với định dạng được chọn.
- In ra thời gian thực hiện của hàm để dễ dàng nhận xét, so sánh.

```
def kcluster5(value,choice):
    start_time = time.time()
    picture = Image.open(value)
    picture = np.asarray(picture)
    X = picture.reshape(picture.shape[0] * picture.shape[1], picture.shape[2])
    kmeans_compressed = Kmeans(kcluster_n=5)
    kmeans_compressed.solving(X)
    picture_kcluster = np.array([list(kmeans_compressed.entr[id]) for id in kmeans_compressed.idx])
    picture_kcluster = picture_kcluster.astype("uint8")
    picture_kcluster = picture_kcluster.reshape(picture.shape[0], picture.shape[1], picture.shape[2])
    fig, ax = plt.subplots(1, 2, figsize = (10, 6))
    ax[0].imshow(picture)
    ax[0].set_title('PICTURE WITHOUT COMPRESSED')
    ax[1].imshow(picture_kcluster)
    ax[1].set_title('5 COLOR COMPRESSED')
    for ax in fig.axes:
        ax.axis('on')
    plt.tight_layout()
    compressed_img = Image.fromarray(picture_kcluster)
    compressed_img.save("compressed5"+choice)
    end_time = time.time()
    print('Your picture ',value,' has been compressed as', choice,' file with 5 color and it took ',start_time-end_time,'s')
```

5. Hàm **enter()** dùng để yêu cầu người dùng nhập tên ảnh.

```
def enter():
    value1= input('ENTER THE NAME OF PICTURE THAT YOU WANT TO COMPRESS: ')
    print('Your name of picture is : ',value1)
    print(' ')
    return value1
```

6. Thực hiện **chương trình chính** bằng cách:

- a. Yêu cầu người dùng nhập tên ảnh
- b. Cho phép người dùng chọn định dạng file xuất ra
- c. Thực hiện nén ảnh với số cụm màu tương ứng và xuất ra file

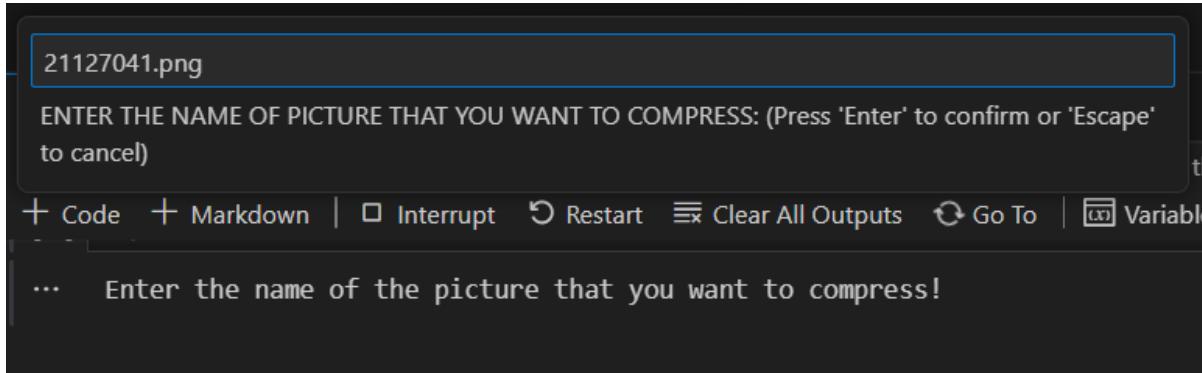
```
print('Enter the name of the picture that you want to compress!')
value=enter()

choice = Formating()
print(' ')
print('ENTER YOUR CHOICE OF OUTPUT FILE FORMAT')
print('Your choice is : ',choice)

kcluster3(value,choice)
kcluster5(value,choice)
kcluster7(value,choice)
kcluster9(value, choice)
```

3.2. Hướng Dẫn Sử Dụng Chương Trình Color Compression:

- **Bước 1:** Chạy chương trình
- **Bước 2:** Nhập vào tên bức ảnh cần compress



```

21127041.png

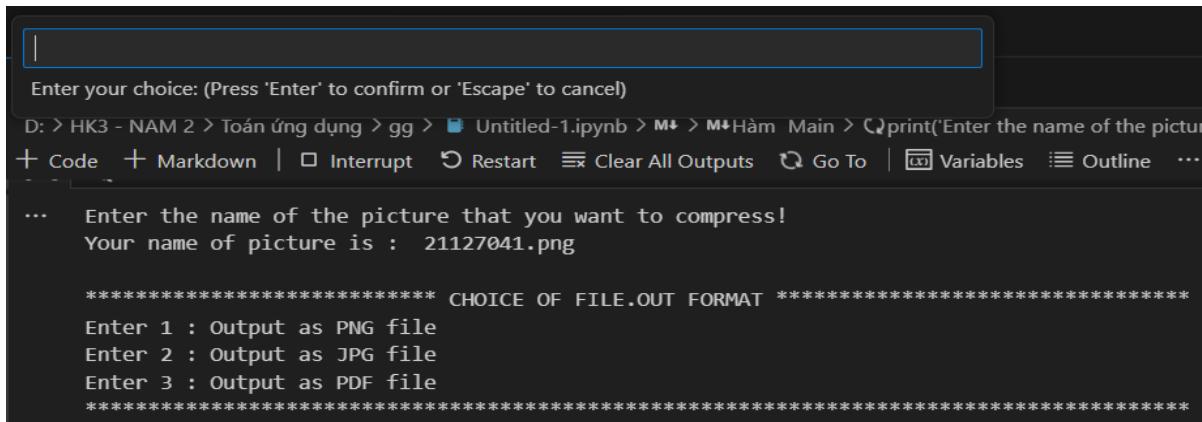
ENTER THE NAME OF PICTURE THAT YOU WANT TO COMPRESS: (Press 'Enter' to confirm or 'Escape' to cancel)

```

+ Code + Markdown | ⏎ Interrupt ⏮ Restart ⏷ Clear All Outputs ⏮ Go To | 📂 Variables

... Enter the name of the picture that you want to compress!

- **Bước 3:** Chọn định dạng đầu ra cho bức ảnh sau khi compressed



```

Enter your choice: (Press 'Enter' to confirm or 'Escape' to cancel)

D: > HK3 - NAM 2 > Toán ứng dụng > gg > Untitled-1.ipynb > M4 > M4 Hàm Main > Qprint('Enter the name of the picture')
+ Code + Markdown | ⏎ Interrupt ⏮ Restart ⏷ Clear All Outputs ⏮ Go To | 📂 Variables 📄 Outline ...
```

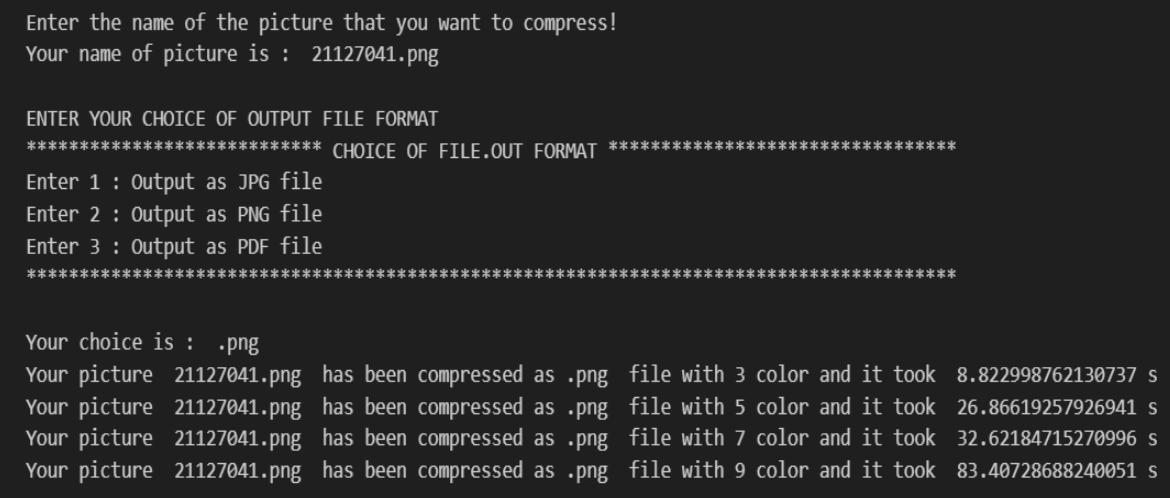
... Enter the name of the picture that you want to compress!
Your name of picture is : 21127041.png

```

***** CHOICE OF FILE.OUT FORMAT *****
Enter 1 : Output as PNG file
Enter 2 : Output as JPG file
Enter 3 : Output as PDF file
*****
```

- **Bước 4:** xem kết quả được hiển thị trên màn hình console, thời gian thực hiện thuật toán và các ảnh được lưu trong folder theo định dạng đã yêu cầu.

- **Với Ảnh có độ phân giải thấp : 953x960**



```

Enter the name of the picture that you want to compress!
Your name of picture is : 21127041.png

ENTER YOUR CHOICE OF OUTPUT FILE FORMAT
***** CHOICE OF FILE.OUT FORMAT *****
Enter 1 : Output as JPG file
Enter 2 : Output as PNG file
Enter 3 : Output as PDF file
*****
```

Your choice is : .png

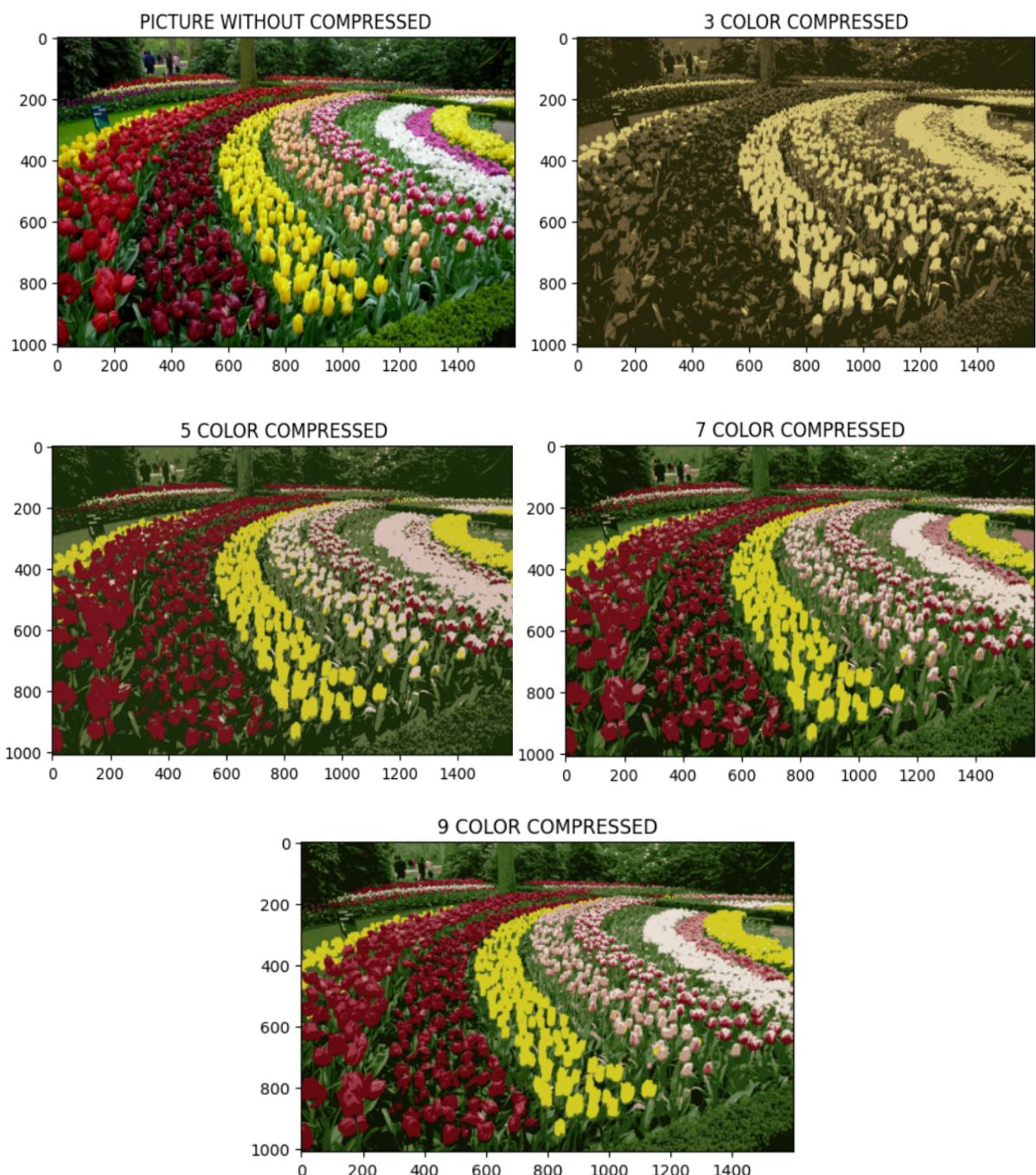
Your picture 21127041.png has been compressed as .png file with 3 color and it took 8.822998762130737 s
Your picture 21127041.png has been compressed as .png file with 5 color and it took 26.86619257926941 s
Your picture 21127041.png has been compressed as .png file with 7 color and it took 32.62184715270996 s
Your picture 21127041.png has been compressed as .png file with 9 color and it took 83.40728688240051 s



○ Với Ảnh có độ phân giải cao : 2119 x 1414

```
Enter the name of the picture that you want to compress!
Your name of picture is : x.png

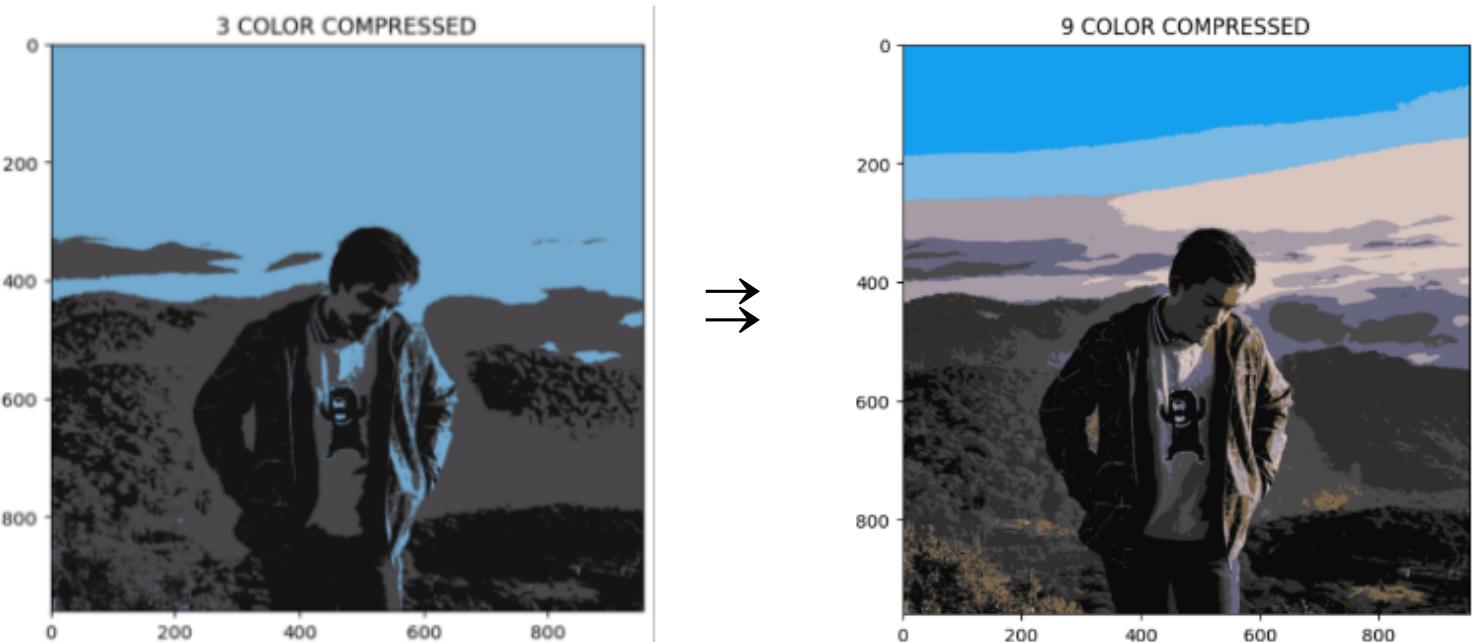
ENTER YOUR CHOICE OF OUTPUT FILE FORMAT
***** CHOICE OF FILE.OUT FORMAT *****
Enter 1 : Output as JPG file
Enter 2 : Output as PNG file
Enter 3 : Output as PDF file
*****
Your choice is : .png
Your picture x.png has been compressed as .png file with 3 color and it took 23.43879270553589 s
Your picture x.png has been compressed as .png file with 5 color and it took 54.53774404525757 s
Your picture x.png has been compressed as .png file with 7 color and it took 136.04597330093384 s
Your picture x.png has been compressed as .png file with 9 color and it took 306.0405795574188 s
```



3.3.Nhận xét kết quả thuật toán:

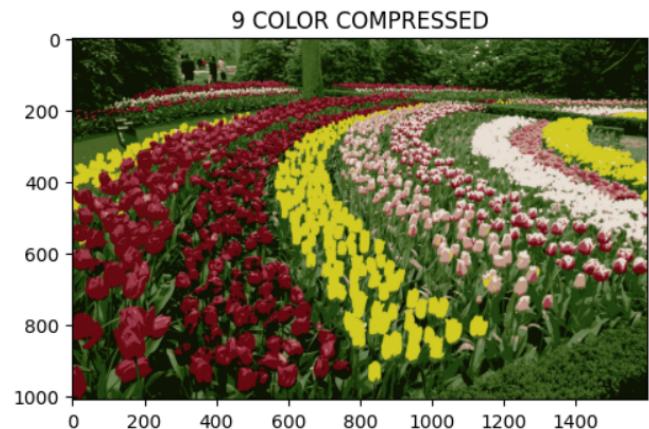
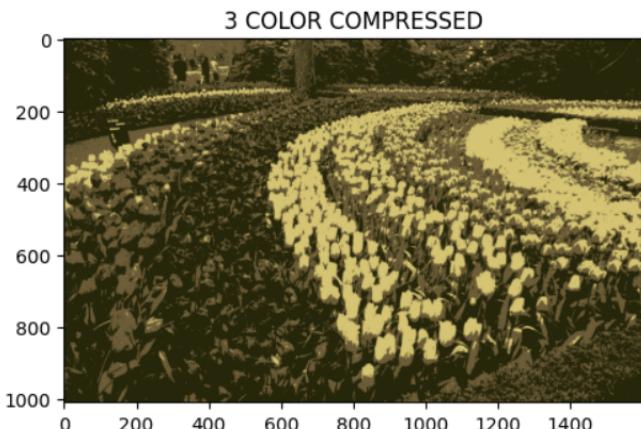
- **ĐỐI VỚI ẢNH CÓ ĐỘ PHÂN GIẢI THẤP 953x960 (ẢNH ĐẦU):**

- Thời gian để nén tấm ảnh **khá nhanh**, chỉ khoảng 8s với 3 màu và **tăng dần khi số lượng màu tăng lên**. Xấp xỉ lần lượt là 26s cho 5 màu, 32s cho 7 màu và 83s cho 9 màu.
- Với k=3, độ chi tiết của tấm ảnh rất thấp, chỉ là hình ảnh 1 người chụp với một background rất mơ màng, không thể hiện được nội dung bức ảnh.
- Lần lượt đến k=5,7 ta có thể hình dung ra được phía sau người này là đồi núi vì đã tăng thêm 2 màu, bức ảnh dần hoàn thiện hơn về nội dung.
- Với k=9, độ chi tiết của bức ảnh được hoàn thiện đáng kể với việc những đồi núi phía sau đã xanh hơn bằng màu của lá cây, của rừng trên đó, tách biệt với màu mây và màu trời.
- Bức ảnh trên có độ phân giải không cao tuy nhiên nó **có nhiều chi tiết, bất cân xứng, không bằng nhau, mờ ảo không rõ ràng** nên trong trường hợp này **K-means không hoạt động tốt** dẫn đến hình ảnh vẫn còn thiếu nhiều chi tiết quan trọng, làm mất đi giá trị cốt lõi của tấm ảnh.



• ĐỐI VỚI ẢNH CÓ ĐỘ PHÂN GIẢI CAO 2119 x 1414 (ẢNH THÚ 2)

- Thời gian để nén tấm ảnh **chậm hơn rất nhiều** so với ảnh có độ phân giải thấp, tận hơn 23s chỉ cho 3 màu và **tăng mạnh khi số lượng màu tăng lên**. Xấp xỉ lần lượt là 53s cho 5 màu, 136s cho 7 màu và 306s cho 9 màu.
- Với các giá trị k tăng dần ($k=3, 5, 7, 9, \dots$), độ chi tiết của bức ảnh hiện lên ngày càng rõ, chất lượng ảnh cải thiện khi k tăng dần.
- Đồng thời, với $K=3$ màu sắc, ta không thể phân biệt được những loại hoa và thậm chí cũng rất khó để phân biệt, đâu là hoa, là lá.
- Với $K=5$, ta thấy bức ảnh có sự thay đổi lớn khi có thể dễ dàng phân biệt giữa lá xanh và hoa, giữa những bông hoa có tông màu khác biệt nhau rõ rệt như đỏ, trắng, vàng.
- Với $K=7, 9$ ta càng phân biệt rõ hơn về màu sắc và chi tiết tấm ảnh vì đã có thể phân biệt được những bông hoa có cùng tông màu nhưng khác sắc độ như luồng hoa đỏ thẫm và luồng màu đỏ tươi, phân biệt được hoa màu vàng và màu hồng phấn. $K=9$ làm cho bức tranh đạt được khá đầy đủ những màu sắc và chi tiết quan trọng mà bức ảnh gốc hướng đến.
- Tấm ảnh trên tuy có độ phân giải rất cao và nhiều chi tiết, tuy nhiên, những **chi tiết trong ảnh, đồng đều, có kích thước tương tự nhau**, các đoá hoa chia đều theo từng luồng với các màu sắc riêng biệt, **không có những vị trí chi tiết mờ ảo, không rõ ràng** nên **thuật toán K-means hoạt động rất tốt** trong tình huống này và cho ra hình ảnh có độ nén cao nhưng vẫn đầy đủ nội dung quan trọng của nó.



CHƯƠNG IV: MỘT SỐ THUẬT TOÁN KHÁC ỦNG DỤNG TRONG COLOR COMPRESSION

4.1. Giới thiệu chung

Color compression là quá trình giảm số lượng màu sắc trong một hình ảnh mà vẫn giữ được độ tương đồng với hình ảnh gốc. Mục đích của việc nén màu là giảm kích thước của hình ảnh, giảm dung lượng của tệp hình ảnh và tăng tốc độ tải hình ảnh trên các trình duyệt web.

Có rất nhiều phương pháp color compression phổ biến:

- **Principal Component Analysis (PCA):** Đây là một thuật toán giảm chiều dữ liệu, được sử dụng để tìm ra các thành phần chính của dữ liệu. Khi color compression, các thành phần chính của các màu sắc trong hình ảnh được tìm ra, và các màu sắc được biểu diễn bằng một số lượng thành phần chính nhỏ hơn. Kết quả là một hình ảnh có số lượng màu sắc giảm xuống.
- **Median Cut:** Đây là một thuật toán mà các màu sắc trong hình ảnh được phân loại vào các nhóm dựa trên khoảng cách trong không gian màu sắc. Các nhóm này sau đó được chia đến khi đủ nhỏ. Kết quả là một bảng màu được tạo ra, và các điểm ảnh trong hình ảnh được thay thế bằng màu sắc tương ứng trong bảng màu.
- **Neural Networks:** Đây là một lớp các thuật toán machine learning được sử dụng rộng rãi trong nhiều bài toán. Trong trường hợp này, một mô hình mạng neural được huấn luyện để học cách nén màu sắc trong hình ảnh. Kết quả là một bảng màu được tạo ra, và các điểm ảnh trong hình ảnh được thay thế bằng màu sắc tương ứng trong bảng màu.
- **Octree quantization:** Đây là một thuật toán color compression, được sử dụng để giảm số lượng màu sắc trong một hình ảnh. Điểm mạnh của thuật toán này là nó có thể giảm số lượng màu sắc một cách hiệu quả mà không ảnh hưởng đến chất lượng hình ảnh.

4.2. Principal Component Analysis (PCA)

4.2.1. Thuật Toán:

Thuật toán PCA được sử dụng trong bài toán **color compression** có cơ chế hoạt động không quá phức tạp đó là giảm chiều dữ liệu bằng cách tìm ra các thành phần chính của dữ liệu, đó là các biến mới có thể giải thích được phương sai của dữ liệu ban đầu một cách tốt nhất. Sau đó, dữ liệu ban đầu được chuyển đổi sang hệ thống tọa độ mới. Việc sử dụng các thành phần chính này có thể giúp giảm chiều dữ liệu và giảm số lượng biến cần lưu trữ. Các bước để thực hiện cũng khá đơn giản:

- **Chuẩn hóa dữ liệu:** Đầu tiên, tất cả các điểm ảnh trong hình ảnh được chuẩn hóa để đảm bảo rằng các giá trị RGB có cùng tỷ lệ và độ lệch chuẩn.
- **Tính toán ma trận hiệp phương sai:** Ma trận hiệp phương sai được tính toán từ các điểm ảnh đã được chuẩn hóa.
- **Tính toán các thành phần chính:** Các thành phần chính được tính toán bằng cách tìm ra các vector riêng của ma trận hiệp phương sai.
- **Lựa chọn số lượng thành phần chính:** Số lượng thành phần chính được lựa chọn sao cho tỷ lệ phương sai tích lũy đạt được một ngưỡng nhất định.
- **Chuyển đổi dữ liệu:** Cuối cùng, tất cả các điểm ảnh trong hình ảnh được chuyển đổi sang hệ thống tọa độ mới, được xác định bởi các thành phần chính.
- **Tạo bảng màu:** Bảng màu được tạo ra bằng cách chọn các màu sắc từ các điểm ảnh đã được chuyển đổi, sao cho số lượng màu sắc được giảm xuống theo số lượng thành phần chính đã chọn.
- **Thay thế các điểm ảnh:** Cuối cùng, các điểm ảnh trong hình ảnh được thay thế bằng màu sắc tương ứng trong bảng màu. Kết quả là một hình ảnh có số lượng màu sắc giảm xuống.

4.2.2. Ưu điểm và Khuyết điểm của thuật toán PCA:

- **Ưu điểm:**

- **Giảm chiều dữ liệu:** PCA giúp giảm số lượng biến trong dữ liệu mà vẫn giữ được thông tin quan trọng. Điều này giúp giảm thời gian tính toán và chi phí lưu trữ dữ liệu.
- **Tính toán đơn giản:** PCA sử dụng các phép tính đơn giản và có thể hiệu quả khi áp dụng cho dữ liệu lớn.
- **Giảm tác động của nhiễu:** PCA có thể giảm tác động của nhiễu trong dữ liệu bằng cách loại bỏ các thành phần chính không quan trọng.

- **Khuyết điểm:**

- **Không thể xác định mối quan hệ nhân quả giữa các biến:** PCA chỉ giúp phân tích quan hệ giữa các biến một cách thống kê và không thể xác định được mối quan hệ nhân quả giữa chúng.
- **Dễ bị ảnh hưởng bởi giá trị ngoại lai:** PCA dễ bị ảnh hưởng bởi giá trị ngoại lai trong dữ liệu, dẫn đến việc các thành phần chính không phù hợp với dữ liệu.
- **Cần chọn số lượng thành phần chính:** Việc chọn số lượng thành phần chính phù hợp là rất quan trọng và có thể ảnh hưởng đến kết quả phân tích của PCA.
- **Không thể áp dụng cho dữ liệu phân loại:** PCA không thể áp dụng cho dữ liệu phân loại, vì nó chỉ hoạt động với dữ liệu số liệu liên tục.

4.3. Thuật Toán Tối Ưu Nhất Cho Bài Toán Color Compression

Hiện nay, có nhiều thuật toán được sử dụng để giải quyết bài toán **color compression** (giảm số lượng màu sắc trong một hình ảnh), tuy nhiên **không có thuật toán nào là tối ưu nhất**, mà phải tùy thuộc vào yêu cầu của từng ứng dụng cụ thể.

Khi số lượng màu sắc trong hình ảnh lớn và đa dạng, khi không cần giữ lại các chi tiết nhỏ trong hình ảnh và khi thời gian tính toán là quan trọng thì ta nên sử dụng thuật toán **K-means Clustering** để giải quyết bài toán color compression

Nếu mục tiêu là giảm số lượng màu sắc trong hình ảnh một cách hiệu quả mà không ảnh hưởng đến chất lượng hình ảnh, **Octree quantization** có thể là một lựa chọn tốt.

Nếu mục tiêu là giảm chiều dữ liệu cho phân tích hoặc mô hình hóa, thì **Principal Component Analysis (PCA)** có thể là một lựa chọn tốt.

Vì vậy, cần phải xem xét cẩn thận các yêu cầu và điều kiện cụ thể trước khi lựa chọn thuật toán phù hợp.

CHƯƠNG V: TÀI LIỆU THAM KHẢO

1. [K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks | by Imad Dabbura | Towards Data Science](#)
2. [The Ultimate Guide to K-Means Clustering: Definition, Methods and Applications](#)
3. [Python PCA using SKLearn for Image Compression Application](#)
4. [Image compression using K-means clustering - GeeksforGeeks](#)
5. [Pros and Cons of K-means Clustering](#)