**ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH**

**ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**


# BÁO CÁO LAB 1

# COLOR COMPRESSION

**HỌ TÊN SINH VIÊN** : **Võ Quốc Bình**

**MÃ SỐ SINH VIÊN** : **21127233**

**LỚP** : **21CLC08 - Toán ứng dụng và thống kê**

**GIÁO VIÊN HƯỚNG DẪN** : **Phan Thị Phương Uyên**

**Thành phố Hồ Chí Minh, tháng 7 năm 202**

# Mục lục

# I. Thông tin sinh viên

- Họ tên sinh viên: Võ Quốc Bình
- Mã số sinh viên: 21127233
- File nộp bài:
    - File code: 21127233.py
    - File báo cáo: 21127233.pdf

# II. Ý tưởng thực hiện

- Ý tưởng của color compression là sử dụng thuật toán K-means để chia các pixels gồm nhiều màu sắc thành một cụm màu nhất định(centroids) từ đó giảm chi phí lưu trữ. Qua đó giảm số lượng màu sắc đồng thời giảm kích thước của hình ảnh mà vẫn giữ được cấu trúc chính của hình.
- Các bước cơ bản cần thiết để xử lý:
    - Tạo centroids bằng 2 cách là random màu sắc và chọn màu bất kì trong các pixels
    - Biến ma trận lưu trữ gồm 3 thành phần (chiều dài, chiều rộng và màu) của từng pixels thành 2 thành phần(các pixels, màu)
    - Hội tụ màu sắc qua từng số lần lặp tối đa(iterations) từ đó update centroids để giúp cho màu sắc giống với nguyên bản hơn
    - Sau đó biến ma trận lưu trữ thành ma trận gồm 3 thành phần như ban đầu và xuất ra tệp file có định dạng jpg/png/pdf như người dùng yêu cầu

# III. Mô tả các hàm

- Ta sử dụng 2 thư viện chính để viết chương trình này numpy(xử lí các phép tính trên ma trận) và PIL(để nhập xuất ảnh)
- Ta có tổng cộng 8 hàm trong chương trình.

## 1. Hàm input_data

- Tham số đầu vào: Không
- Giá trị trả về: filename, k_clusters, iteration và type_centroids(kiểu chọn centroids)
- Mô tả:
    - Đầu tiên cho người dùng nhập filename, nếu định dạng sai(khác png và jpg) thì cho người dùng nhập lại.
    - Tương tự cho người k_clusters và iteration.
    - Nhập type_centroids: cho người dùng chọn 2 loại là “random” và “in_pixels”, nếu người dùng nhập sai thì mặc định cho người dùng nhập vào là “in_pixels”.

## 2. Hàm output_data

- Tham số đầu vào: filename, k_clusters, outputImg
- Giá trị trả về: Không
- Mô tả:
    - Cho người dùng nhập định dạng mà người dùng muốn xuất ra(jpg/png/pdf), nếu sai thì nhập lại cho đến khi đúng.
    - Xuất ra file với định dạng: {filename}_k{k_clusters}.{tên định dạng}
    - Sử dụng thư viện PIL với hàm formarray đề phục hồi lại ảnh từ ma trận numpy đã nhập vào(outputImg)

## 3. Hàm init_centroid

- Tham số đầu vào: flat_img, k_cluster, type_centroid
- Giá trị trả về: centroids(số clusters được tạo bằng random hoặc in pixel)
- Mô tả:
    - Tạo ma trận 0 với k hàng và 3 cột tương ứng với số clusters và 3 thông số RGB để tạo nên màu sắc
    - Nếu người dùng chọn type_centroid = “random” thì random 3 thông số RGB đồng thời đảm bảo trước khi bỏ màu vào ma trận đã tạo thì không có ma trận

trùng lặp nếu trùng lặp thì ta thực hiện random lại cho đến khi các màu khác nhau

- Nếu người dùng chọn type_centroid = “in_pixels” thì ta chọn màu từ ma trận flat_img để chọn màu bất kì và bỏ vào ma trận đã tạo. Nếu màu đó đã xuất hiện thì t thực hiện chọn lại màu cho đến khi khác màu đã có trong ma trận.

## 4. Hàm labels_img

- Tham số đầu vào: flat_img, centroids
- Giá trị trả về: labels
- Mô tả:
  - Trước đó ta tạo một ma trận mới để lưu các giá trị label này.
  - Ta tiến hành tính toán khoảng cách giữa flat_img và centroids đã tạo
  - Giá trị khoảng cách nào mà nhỏ nhất thì là gán giá trịn centroid đó vào label rồi trả về cho hàm.

## 5. Hàm update_centroids

- Tham số đầu vào: flat_img, labels_arr, centroids_info
- Giá trị trả về: new_centroids
- Mô tả:
  - Tạo ma trận 0 đề lưu giá trị centroids mới
  - Lặp hết qua các centroids, lấy các pixel thuộc về cluster i bằng cách sử dụng mảng labels_arr. Cụ thể, labels_arr == i tạo ra một mảng boolean với giá trị True tại vị trí mà nhãn tương ứng của pixel là i. Sau đó, flat_img[labels_arr == i] lấy các pixel tương ứng với các vị trí True này và gán cho pixel_cluster.
  - Kiểm tra xem cụm i có chứa ít nhất một pixel hay không. Nếu có, tiếp tục với các bước tiếp theo để tính toán centroid mới.
  - Ta tính toán bằng cách tính toán giá trị trung bình của các pixel trong cụm i để tạo ra centroid mới. Hàm mean của numpy dùng để tính toán các giá trị trung bình của các cột “pixel_cluster”

## 6. Hàm kmeans

- Tham số đầu vào: flat_img, k_clusters, max_iteration, type_centroids
- Giá trị trả về: centroids, labels
- Mô tả:
  - Tạo ma trận 0 giống với flat_img đề lưu giá trị các các labels và khởi tạo centroid cho thuật toán K-means

- Lặp hết max_iteration. Trong lúc lặp ta liên tục update giá trị labels và centroids đề có được màu sắc chính xác nhất
- Trong lúc lặp, nếu nhận thấy centroid cũ bằng với centroid vừa update thì ta thoát khỏi vòng lặp, ta cho giá trị xấp xỉ là 10e-5

## 7. Hàm change_2d_to_1d

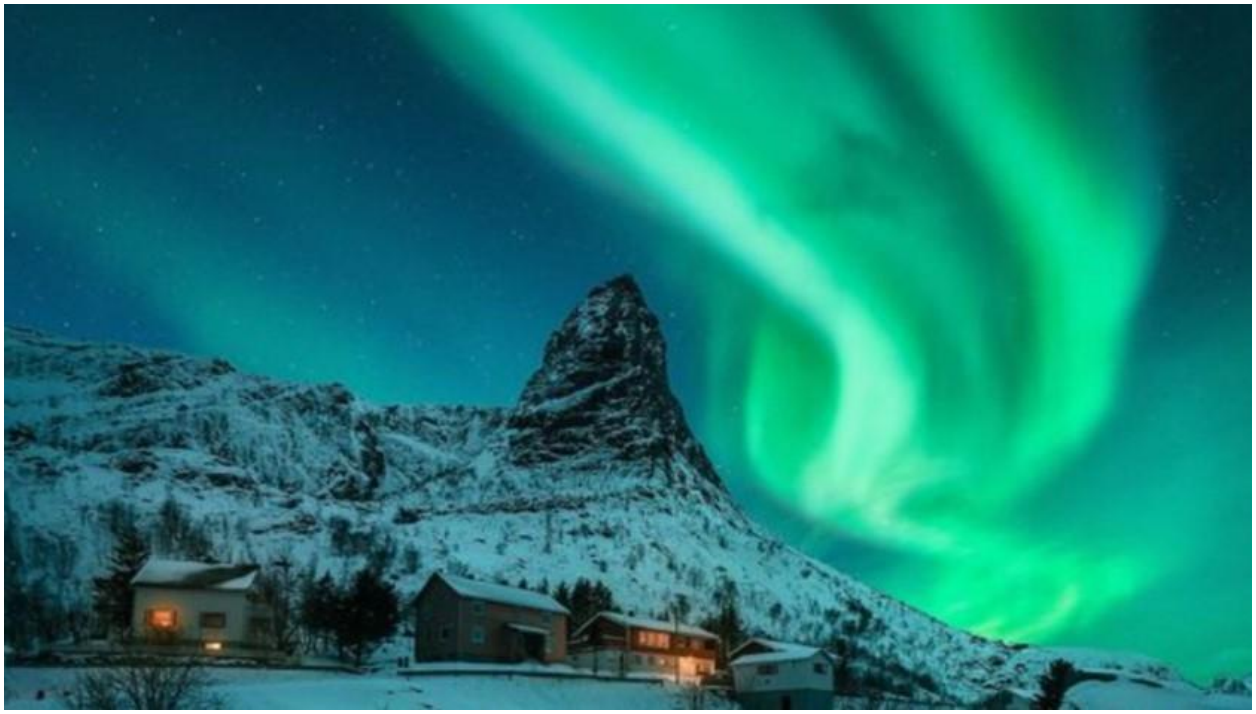
- Tham số đầu vào: filename
- Giá trị trả về: flatImage, image
- Mô tả:
  - Mở hình ảnh với file đầu vào với thư viện PIL
  - Chuyển hình ảnh vừa lấy được thành hình ảnh với thư viện numpy
  - Ta chuyển thành ma trận 2 chiều bằng cách nhân chiều dài và chiều rộng với hàm reshape của numpy

## 8. Hàm change_1d_to_2d

- Tham số đầu vào: centroids, labels, image
- Giá trị trả về: Hình ảnh sau khi được reshape

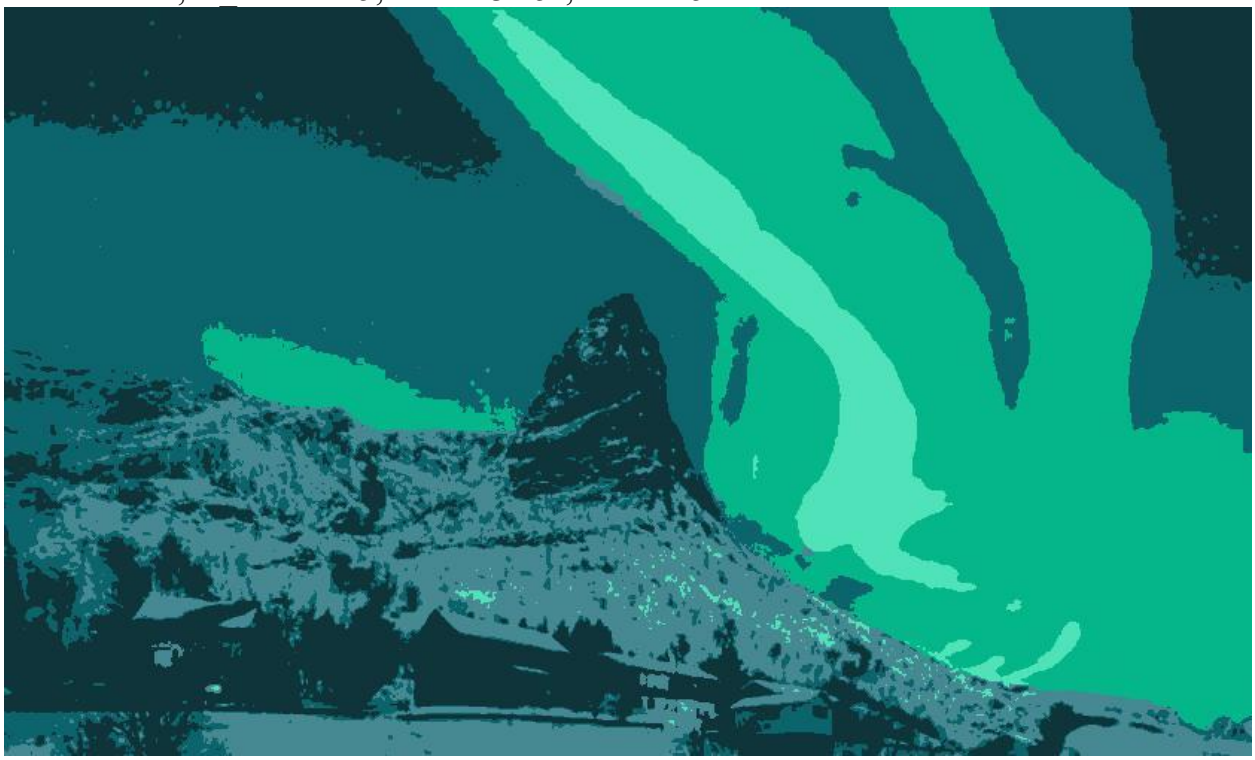
# IV. Nhận xét ảnh sau khi qua xử lí

- Ảnh đầu vảo (size = 70KB):



- Ta thực hiện với max_iteration = 1000
- Random, K_cluster =3, time = 3.41s, size = 44 KB

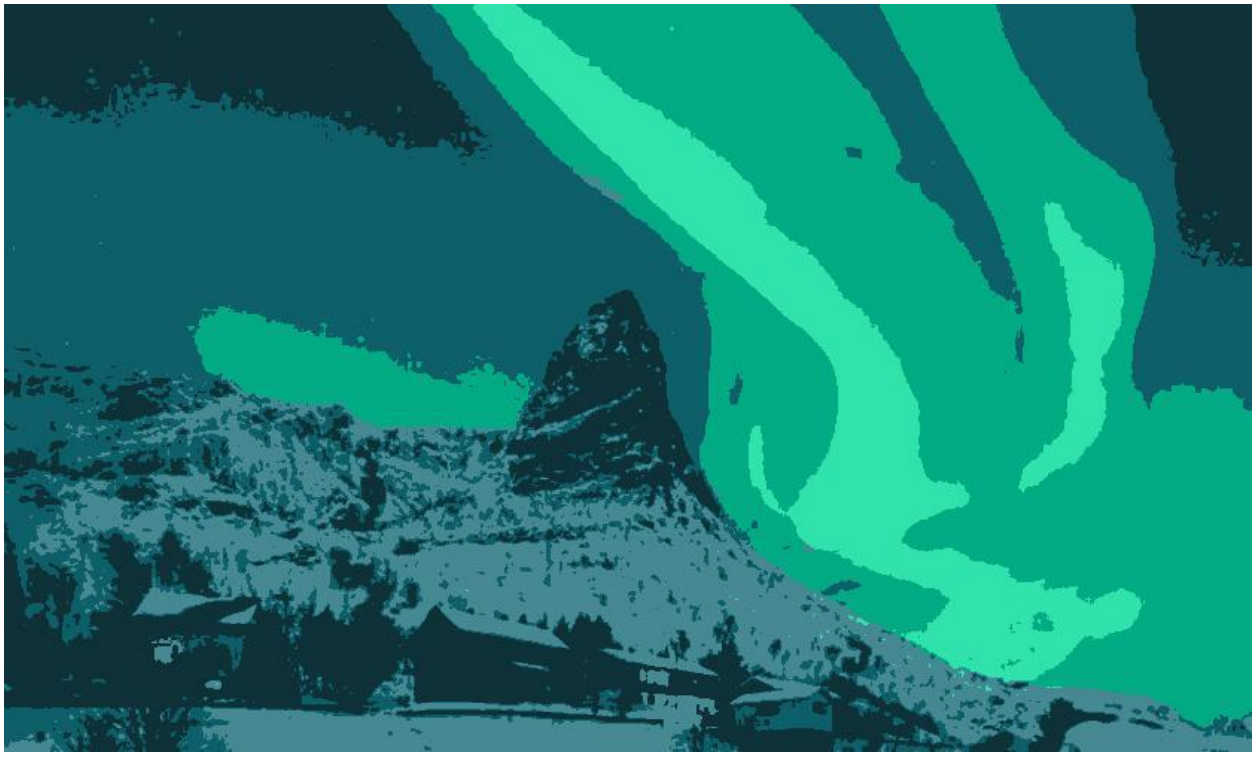
- Random, K_cluster =5, time = 3.15s, size = 45 KB



- Random, K_cluster =7, time = 5.99s, size =47 KB

- In pixel, K_cluster = 3, time = 0.66s, size = 39 KB

- In pixel, K_cluster = 5, time = 5.4s, size = 44 KB



-

- In pixel, K_cluster = 3, time = 9,2s, size =47 KB

⇨ Nhận xét:
- Số lượng màu giảm đi đáng kể nhưng vẫn giữ được nội dung
- Giảm thiểu dung lượng cho bức ảnh

# V. Tham khảo


- https://github.com/shuyangsun/k_means_clustering
- https://www.youtube.com/watch?v=shu4pYQb-ps - Image compression & Color Quantization using K Means | k-means Clustering in Unsupervised ML