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 THỰC HÀNH

MÔN: TOÁN ỨNG DỤNG VÀ THỐNG KÊ (MTH00057 – 21CLC01)

LAB01: KMEANS

GIẢNG VIÊN LÝ THUYẾT NGUYỄN ĐÌNH THÚC

GIẢNG VIÊN THỰC HÀNH NGUYỄN VĂN QUANG HUY
NGÔ ĐÌNH HY

SINH VIÊN THỰC HIỆN 21127621 – ÂU DƯƠNG KHANG

MỤC LỤC

1) Cài đặt

Thuật toán Kmeans làm giảm số lượng màu trong một bức ảnh với số lượng màu K cho trước (K cluster) nhằm giảm chi phí lưu trữ ảnh xuống so với ảnh ban đầu mà vẫn bảo toàn nội dung ảnh nhất có thể. Ý tưởng thuật toán được đưa ra như sau:

1. Chọn K điểm bất kỳ làm các center ban đầu.
2. Phân mỗi điểm dữ liệu vào cluster có center gần nó nhất.
3. Nếu việc gán dữ liệu vào từng cluster ở bước 2 không thay đổi so với vòng lặp trước nó thì ta dừng thuật toán.
4. Cập nhật center cho từng cluster bằng cách lấy trung bình cộng của tất các các điểm dữ liệu đã được gán vào cluster đó sau bước 2.
5. Quay lại bước 2.

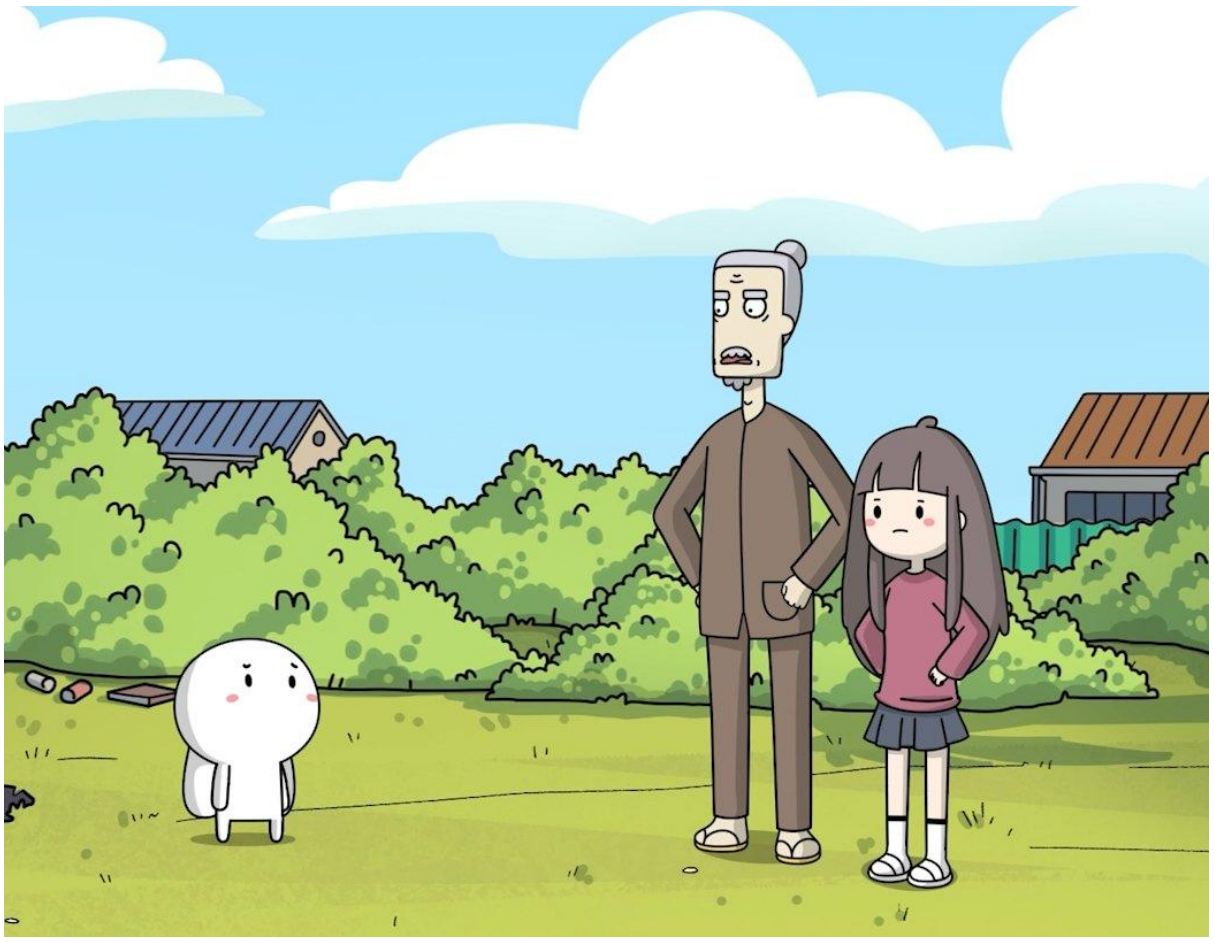
Để làm được điều đó cần có các hàm chức năng như sau:

- `def init_centroid(image_1d, K, type):` Tạo ra tập các center (centroid) tùy thuộc vào số cluster và loại centroid đã nhập. Có 2 loại centroid là random, tạo ra tập các center bằng các con số ngẫu nhiên nằm trong khoảng [0, 255], và in_pixel, tạo ra tập các center bằng cách lấy ngẫu nhiên các điểm trong ảnh. Centroid được tạo ra có kích thước (số cluster x số cột của ảnh sau khi reshape).
    + INPUT: ảnh đã reshape, số cluster K, loại centroid muốn tạo (random hoặc in_pixel)
    + OUTPUT: một mảng các center
- `def compute_euclidean_distance(point, centroid):` Tính khoảng cách euclid giữa các điểm với các center.
    + INPUT: điểm của ảnh, tập các center (centroid)
    + OUTPUT: mảng kết quả khoảng cách của điểm tới các center
- `def assign_label(image_1d, centroid):` Dán nhãn cho các điểm trong ảnh. Cách để dán nhãn các điểm trong ảnh là tính khoảng cách các điểm so với các center, kết quả được lưu lại thành 1 mảng có kích thước (1 x số center), sau đó tìm chỉ số của phần tử có giá trị thấp nhất, đó cũng chính là nhãn chúng ta cần tìm.
    + INPUT: các điểm của ảnh, tập các center (centroid)
    + OUPUT: mảng các nhãn ứng với các điểm trong ảnh.
- `def update_centroid(image_1d, K, centroid):` Cập nhật danh sách các center khi các nhãn của điểm thay đổi. Khi các nhãn của điểm thay đổi ta có thể gom những điểm có nhãn giống nhau về cùng 1 cluster, từ đó ta cập nhật center mới bằng cách trung bình cộng các điểm thuộc cluster đó.
    + INPUT: các điểm của ảnh, số cluster K, tập các center (centroid)
    + OUTPUT: tập các center mới (new centroid)
- `def has_converged(centroid, new_centroid):` Kiểm tra điều kiện dừng của thuật toán. Thuật toán chỉ dừng lại khi các center vừa được tính toán xong có giá trị bằng với các giá trị của các center trước đó.
    + INPUT: tập các center trước đó (centroid), tập các center mới (new centroid)
    + OUTPUT: true or false (các giá trị của 2 centroid có bằng nhau hay không)

- `def set_pixel(image_1d, label, centroid):` Đặt lại các giá trị cho các điểm của ảnh ứng với các giá trị của nhãn và giá trị center mới đã được cập nhật. Từ đó ta có được hình ảnh đã được giảm số lượng màu.
    + INPUT: các điểm của ảnh, nhãn của các điểm, tập các center (centroid)
    + OUTPUT: ảnh mới được thay đổi các giá trị để giảm màu.
- `def kmeans(image_1d, k_clusters, max_iter, init_centroids = 'random'):` Hàm chạy thuật toán Kmeans. Ở trong hàm sẽ xử lý dữ liệu input như reshape lại ảnh, đặt loại centroid muốn sử dụng, sau đó thực thi thuật toán trong 1 vòng lặp có giới hạn số lần lặp và đặt lại các giá trị của các điểm ảnh để tạo ra ảnh mới đã giảm số lượng màu theo số K đã nhập.
    + INPUT: ảnh ban đầu được nhập vào, số cluster K, số vòng lặp tối đa cho phép, loại centroid muốn thực hiện
    + OUTPUT: centroid gồm các mảng center trong quá trình thực thi thuật toán, nhãn cuối cùng của các điểm, ảnh được giảm màu.
- `def main:`
    + INPUT: nhập vào hình muốn giảm màu.
    + OUTPUT: hiển thị các center, nhãn sau khi chạy xong thuật toán cùng với hình đã giảm màu dưới 2 định dạng: png và pdf.

2) Danh sách testcase

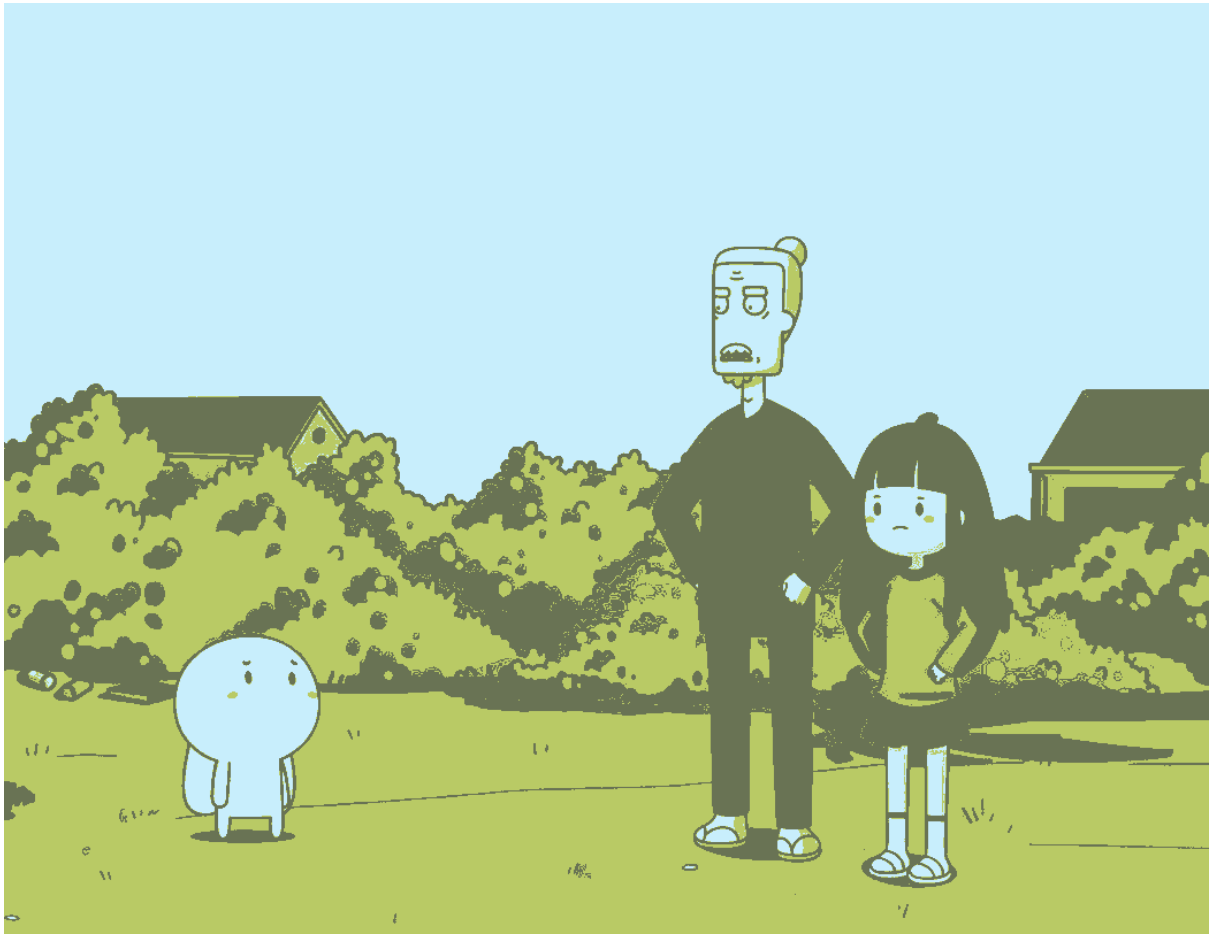
INPUT:



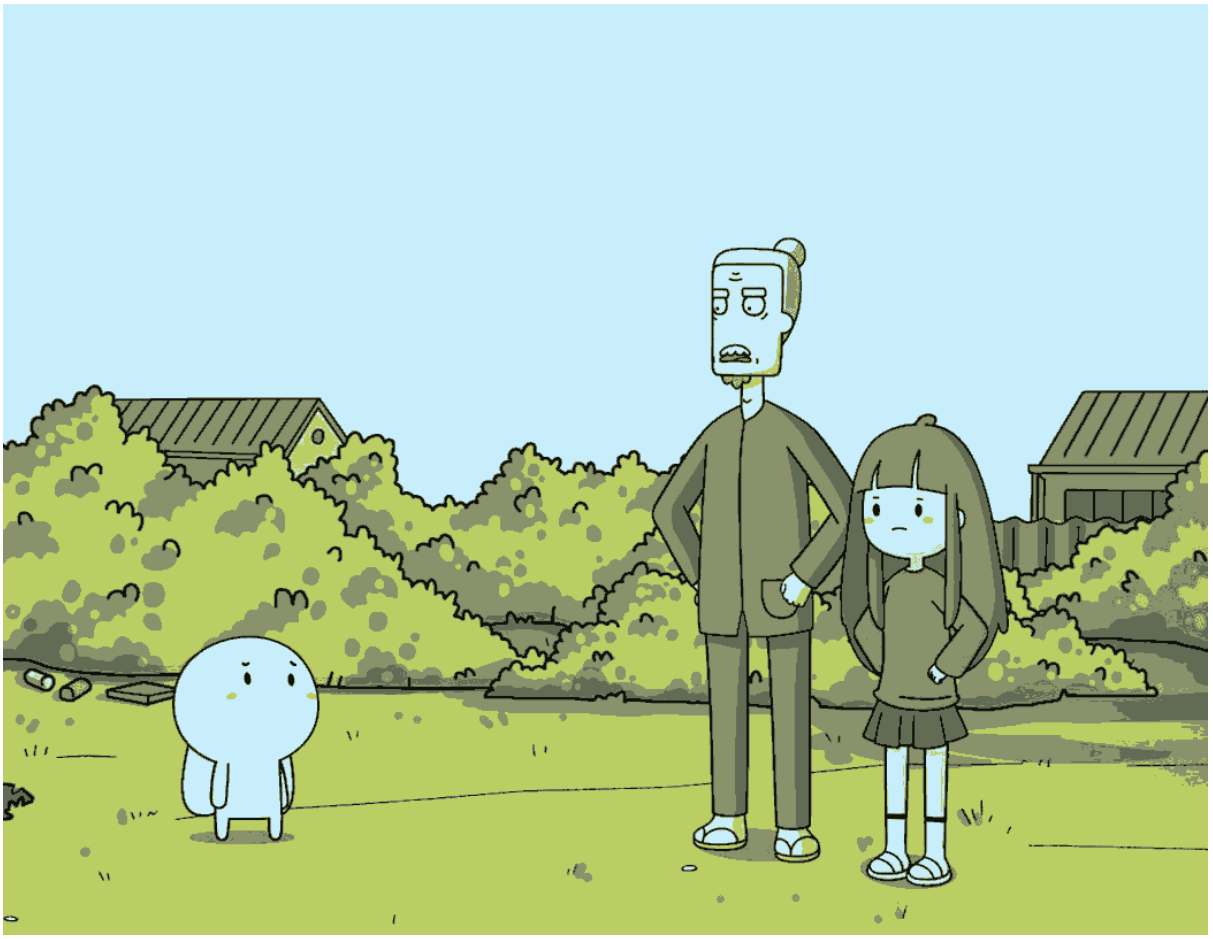
OUTPUT:

- Loại centroid “random”:

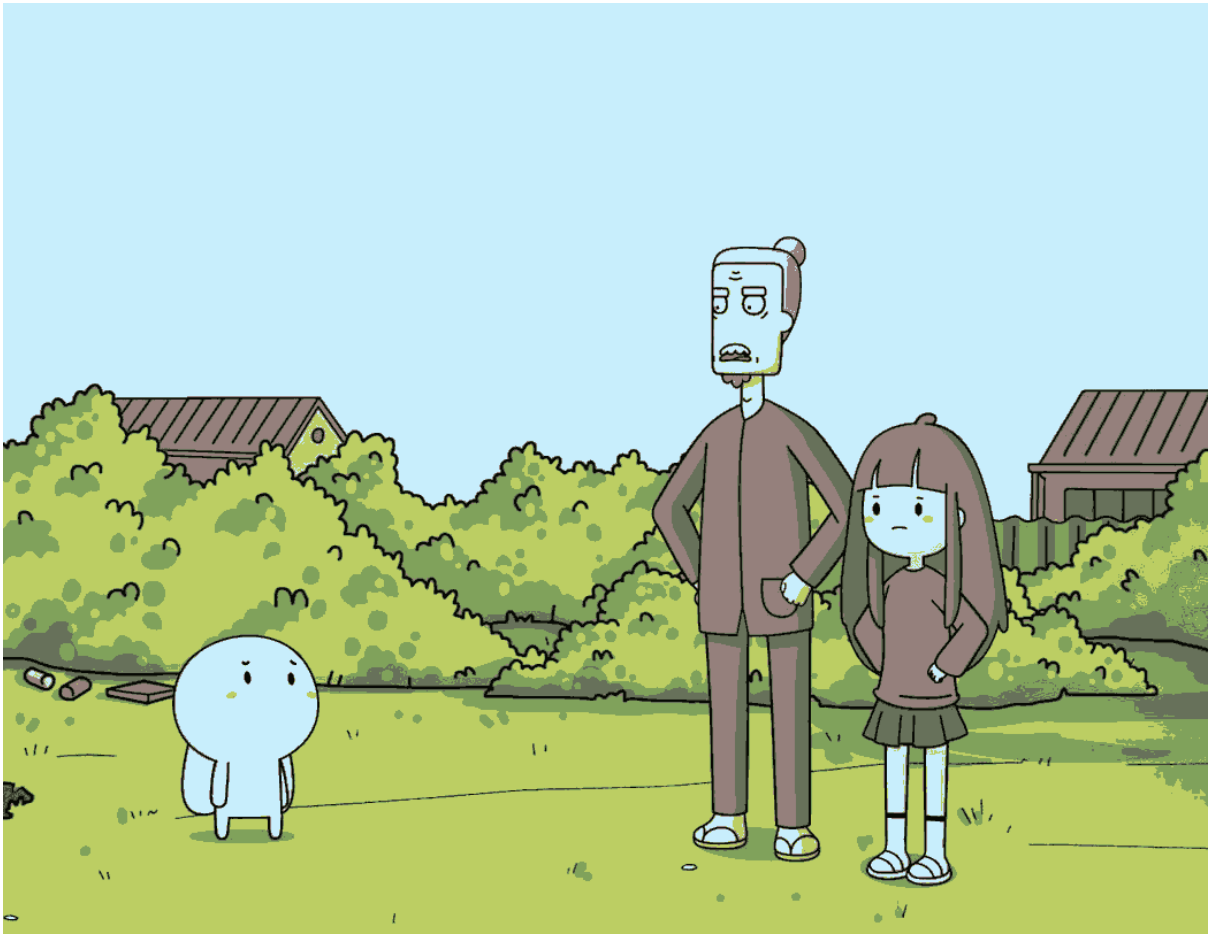
+ K=3



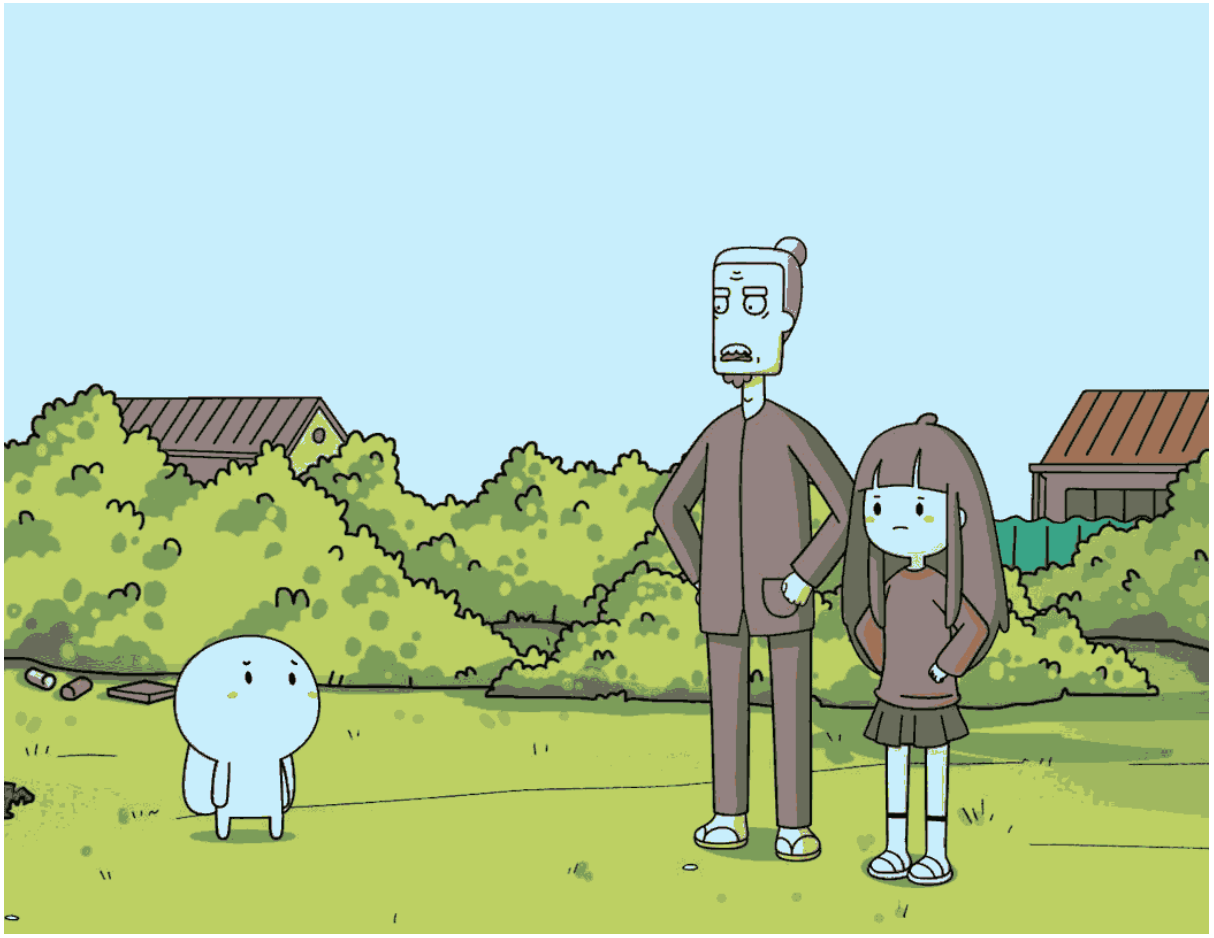
+ K=5



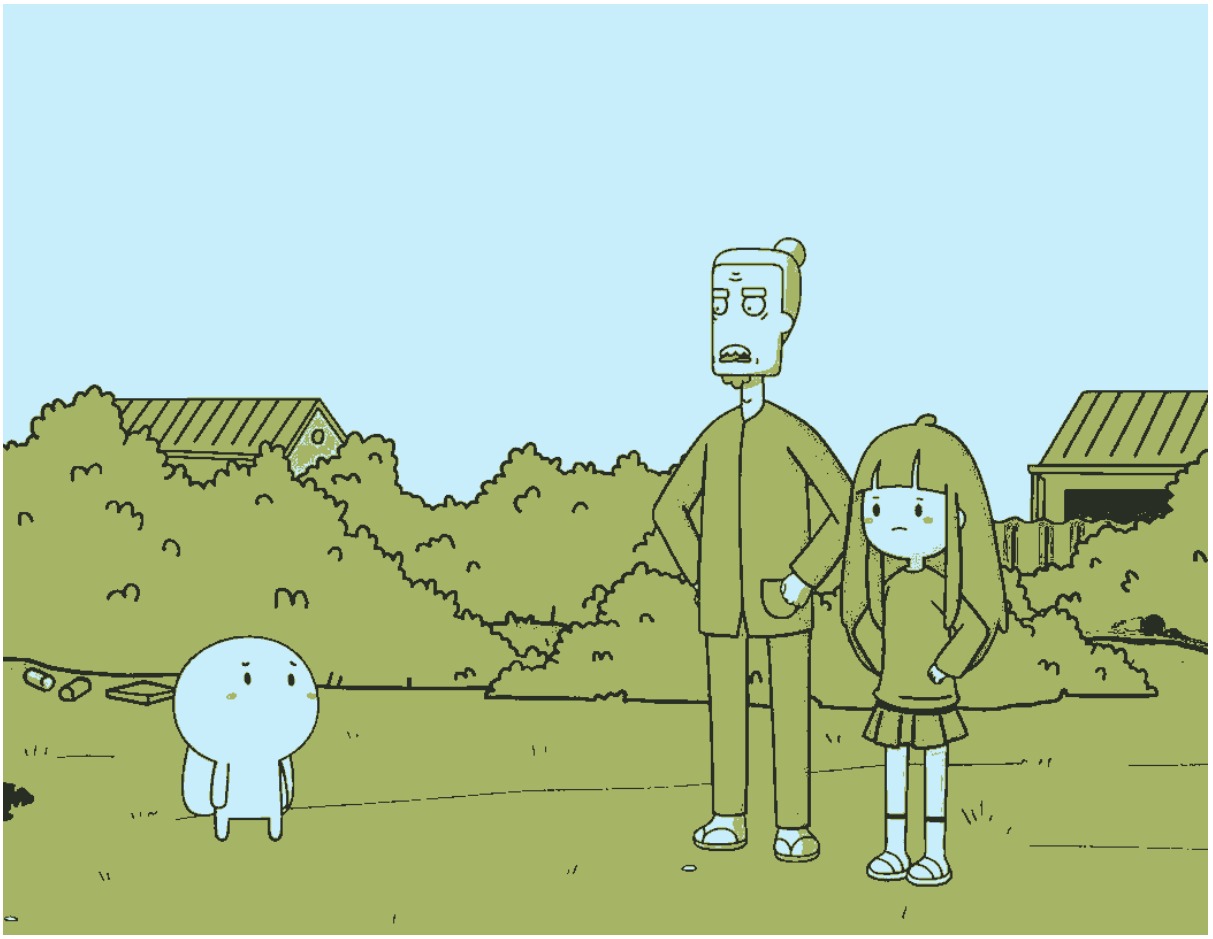
+ K=7



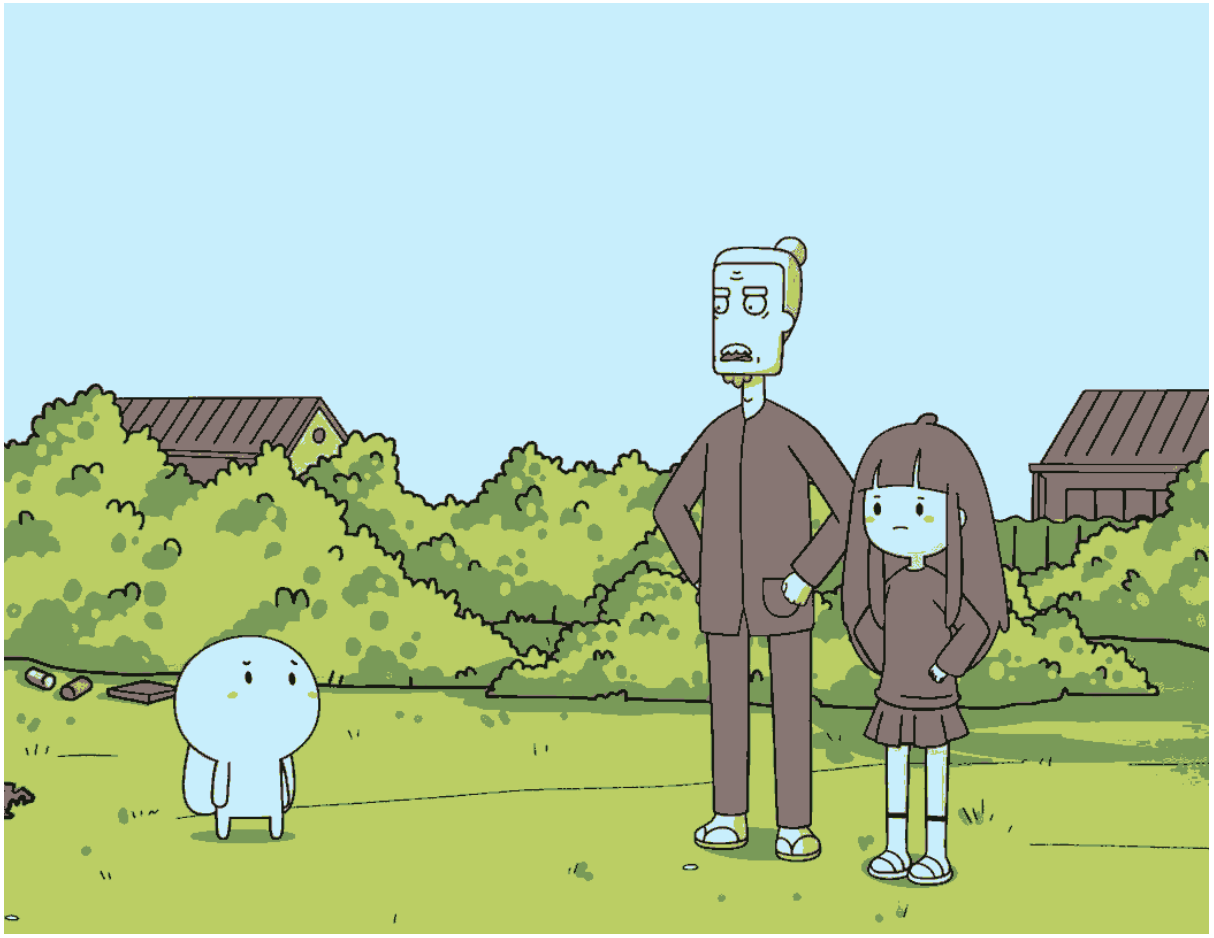
+ K=10



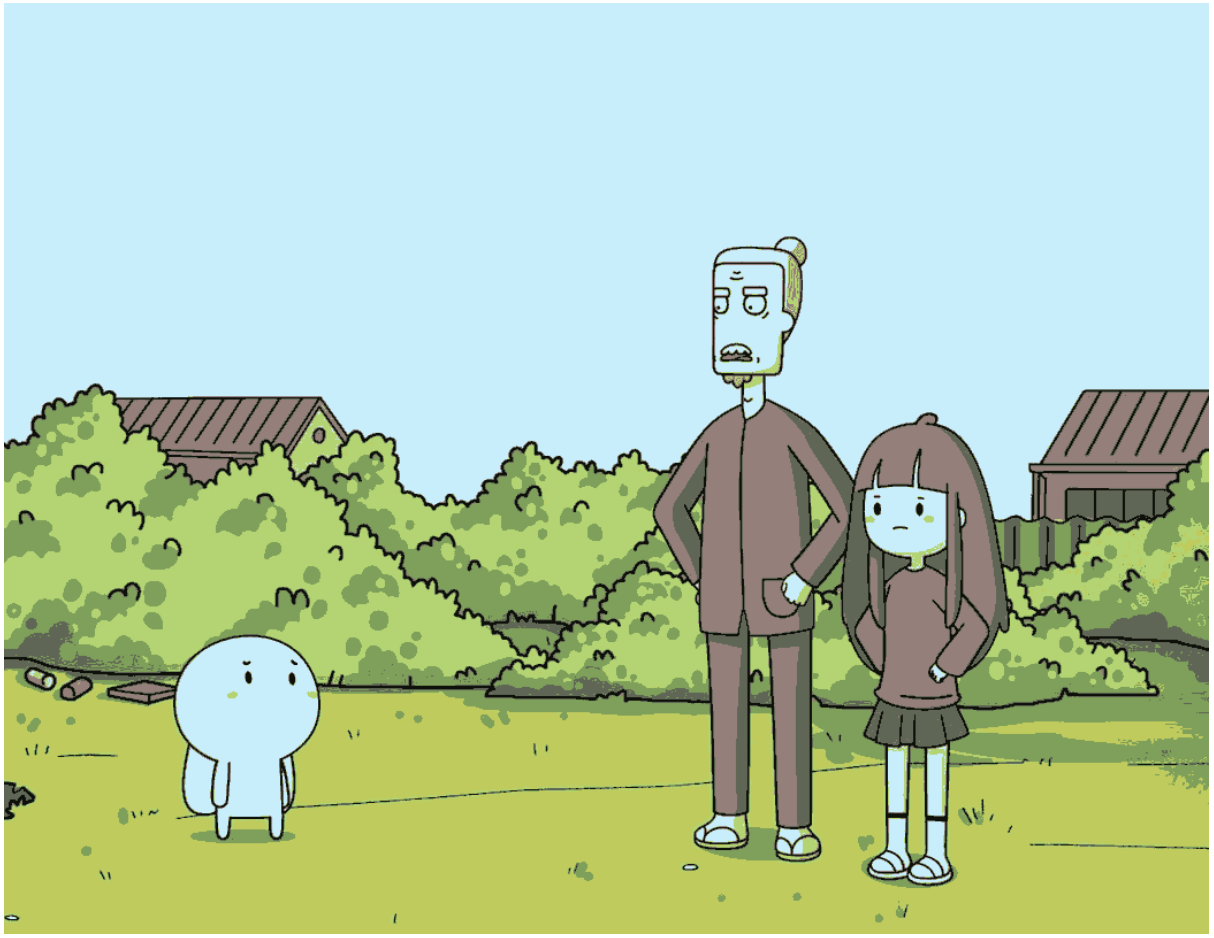
- Loại centroid “in_pixel”:
    + K=3



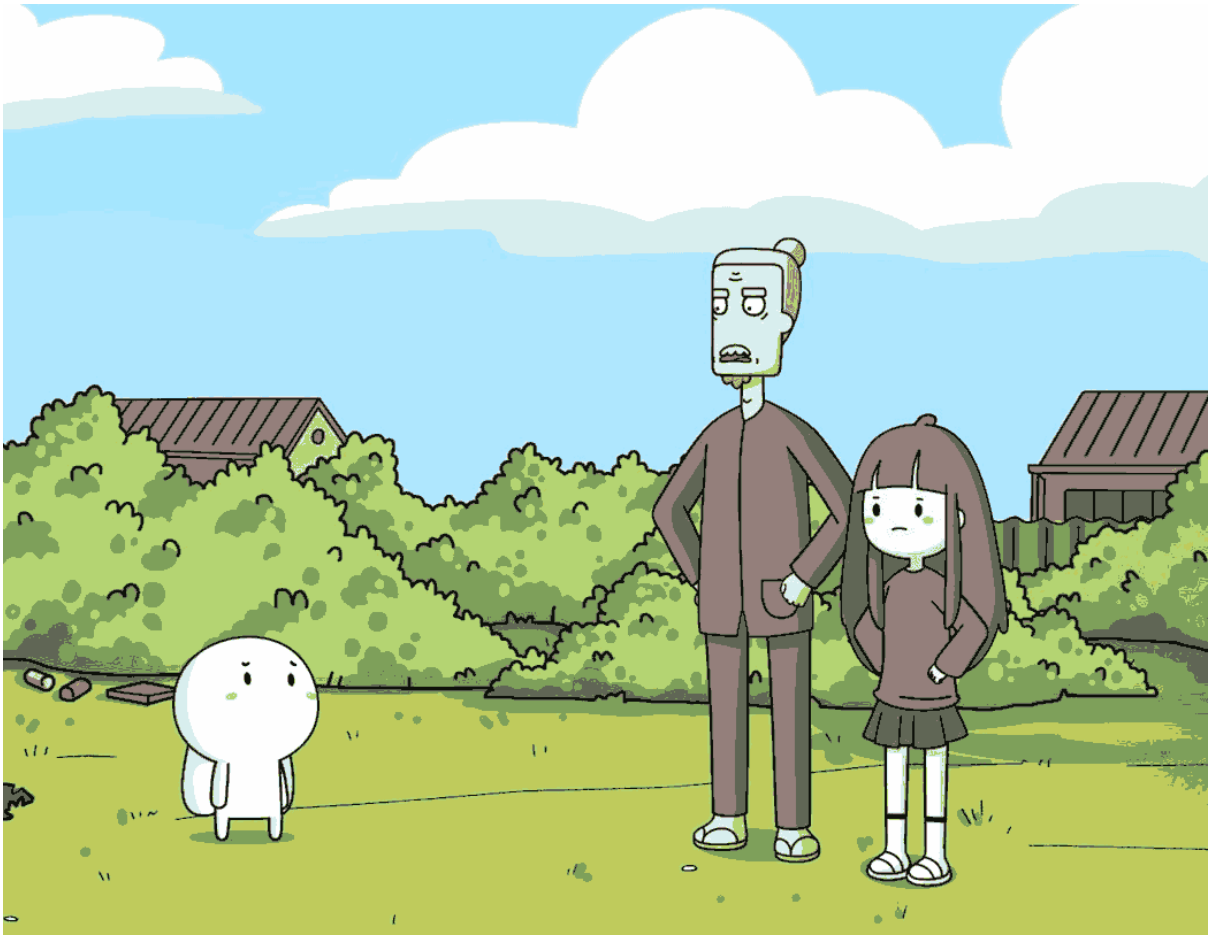
+ K=5



+ K=7



+ K=10



- Ngoài ra còn có output dạng pdf: Kmeans PDF output

3) Đánh giá

- Ưu điểm: Giải thuật cho ra kết quả, không xuất hiện lỗi, giảm màu thành công ảnh đã cho.
- Nhược điểm: Đối với bài toán có input phức tạp (ảnh có nhiều màu, định dạng ảnh lớn), số K lớn thì thuật toán có thể xử lý chậm hay không cho ra kết quả mong muốn. Ngoài ra, thuật toán còn chưa hoàn chỉnh ở các hàm nhỏ (còn tham khảo các tài liệu bên ngoài, sử dụng thư viện còn chưa hợp lý).

4) Nguồn tham khảo

Machine Learning cơ bản - Kmeans
GitHub - Kmeans