Omar Ahmad M146 HW7 #5

5.

a.

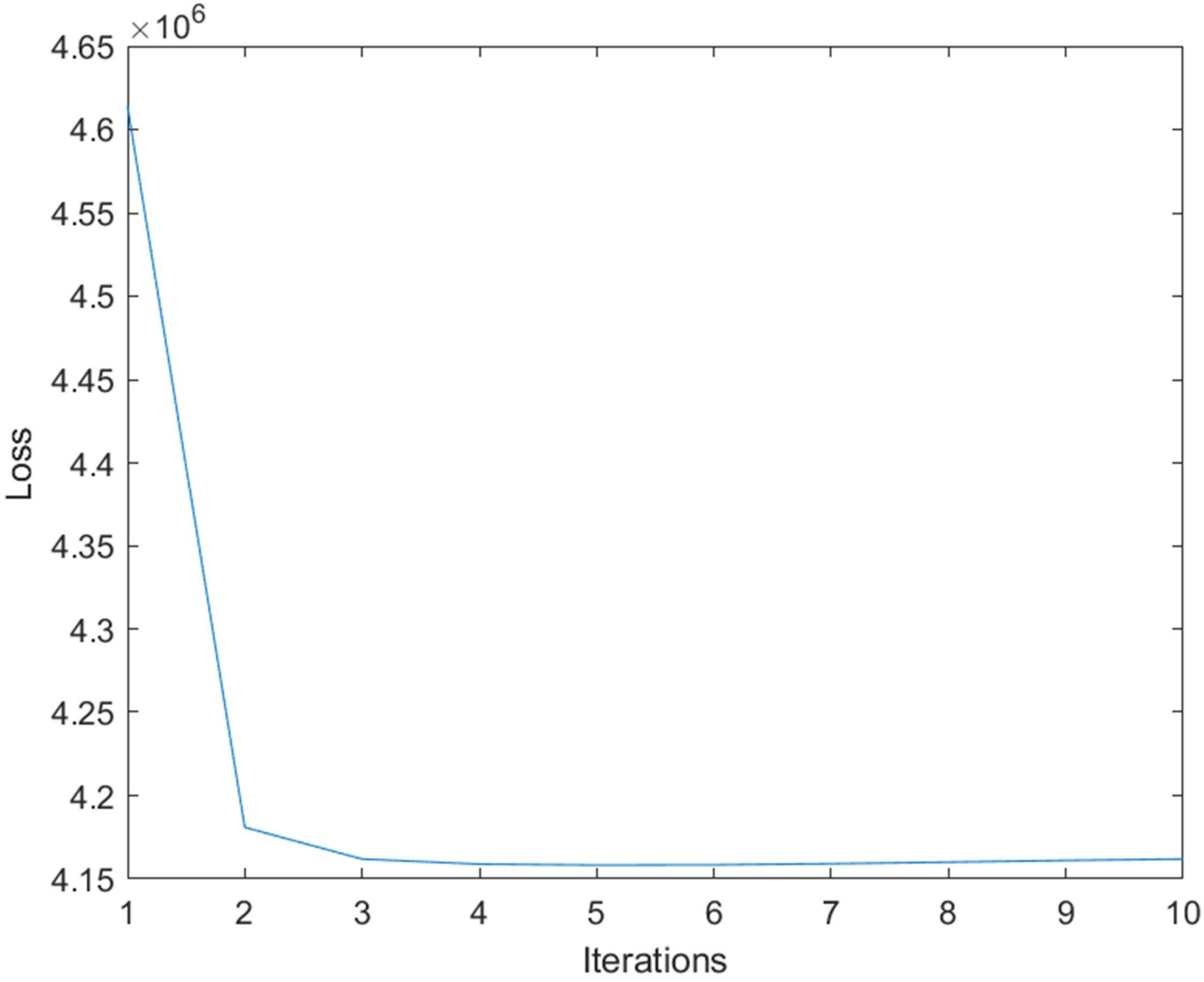
hw7a.m

```
clear;
close;
clc;

img = imread('UCLA_Bruin.jpg');
imshow(img);

```

b.



The algorithm converges after about 4-5 iterations and after that, the loss function doesn't change much.

Code:

```
clear;
close;
%clc;

img = double(imread('UCLA_Bruin.jpg'));
K = 4;
numIter = 10;
u(1,:) = double(squeeze(img(1,1,:)))';
numMeans = 1;

for k = 2:K
    for i = 1:numMeans
        %dist of all pixels to ith mu (u)
        dist(:,:,i) = sqrt((img(:,:,1) - u(i,1)).^2 + (img(:,:,2) - u(i,2)).^2 + (img(:,:,3) - u(i,3)).^2);
    end
    % now know dist of every pixel to every mu (that currently exists)

    % the min dist for each pixel (to the closest u)
    minDists = min(dist,[],3);

    [M,I] = max(minDists);
    [N,J] = max(M); %N is largest elem of minDists
    Z = I(J);
    if(N ~= minDists(Z,J))
        disp("ERROR!");
    end
    %dist(Z,J) is the max dist
    %img(Z,J,:) is the RGB for that pixel

    newMeanIndex = numMeans+1;
    u(newMeanIndex,1) = img(Z,J,1);
    u(newMeanIndex,2) = img(Z,J,2);
    u(newMeanIndex,3) = img(Z,J,3);
    numMeans = numMeans + 1;
end
% now initialized (k mu's set by furthest-first rule)
```

```
%each iter do assignment and re-estimation of center of cluster
%and also calc loss func for tracking the alg
Loss = zeros(1,numIter);
for iter = 1:numIter
    %assignment
    for k = 1:K
        dist(:,:,k) = sqrt((img(:,:,1) - u(k,1)).^2 + (img(:,:,2) - u(k,2)).^2 + (img(:,:,3) - u(k,3)).^2);
    end
    for i = 1:size(img,1)
        for j = 1:size(img,2)
            [M,I] = min(dist(i,j,:));
            assign(i,j) = I;
        end
    end

    %re-estimation
    % pts(i,:) is the sum of the RGB elem of all pts in img w/ assign() = i
    pts = zeros(K,3);
    numPts = zeros(1,4);
    for i = 1:size(img,1)
       for j = 1:size(img,2)
           tmp = assign(i,j);
           pts(tmp,:) = pts(tmp,:) + squeeze(img(i,j,:))';
           numPts(tmp) = numPts(tmp) + 1;
       end
    end
    for k = 1:K
       u(k,:) = pts(k,:)/numPts(k);
    end

    %loss func
    for i = 1:size(img,1)
        for j = 1:size(img,2)
            tmp = assign(i,j);
            Loss(iter) = Loss(iter) + norm(squeeze(img(i,j,:))' - u(tmp,:));
        end
    end
end

plot([1:numIter], Loss);
xlabel("Iterations")
ylabel("Loss")
```

c.

K = 4:

Final value of objective function: 4.1619×10^6

K = 8:

Final value of objective function: 2.827×10^6

K = 16:

Final value of objective function: 1.8979×10^6

- Increasing K increases the quality of the end image. This is because there are more colors that can be used. So for K = 4, each pixel gets assigned to one of 4 colors which will have less detail/quality than if they were assigned to one of 16 colors.

d.
300x400 = 120,000 pixels, each taking 3*8 bits
120,000*24 bits = 2,880,000 bits = 2812.5 Kilo bits (2,880,000/1024)
→**2,880,000 bits to store original image**

Data Compression Ratio = uncompressed size / compressed size

New:
K = 4:
- 2 bits for each pixel to store the index of which cluster = log(base 2)4
- + 8 bits for each cluster * 4 clusters = 32 bits
- 120,000*2 = 240,000 + 32 = 240,032 bits for K = 4
- Compression Ratio = 2,880,000/240,032 = 11.998

K = 8:
- 3 bits for each pixel to store the index of which cluster = log(base 2)8
- + 8 bits for each cluster * 8 clusters = 64 bits
- 120,000*3 = 360,000 + 64 = 360,064 bits for K = 8
- Compression Ratio = 2,880,000/360,064 = 7.999

K = 16:
- 4 bits for each pixel to store the index of which cluster = log(base 2)16
- + 8 bits for each cluster * 16 clusters = 128 bits
- 120,000*4 = 480,000 + 128 = 480,128 bits for K = 16
- Compression Ratio = 2,880,000/480,128 = 5.998

K	Bits	Compression Ratio
4	240,032	11.998
8	360,064	7.999
16	480,128	5.998