```java
    public byte[] histEq(byte[] data, int width, int height){
        byte[] histeqData = new byte[data.length];
        int size = height * width;

        // Perform Histogram Equalization
        // Note that you only need to manipulate data[0:size] that corresponds to luminance
        // The rest data[size:data.length] is for colorness that we handle for you
        // ********************** START YOUR CODE HERE  ************************** //
        int array_len = width *height;
        int[] hist = new int[256];
        int[] cdf  = new int[256];
        int[] h    = new int[256];
        int[] data_= new int[array_len];
        int[] data_o = new int[array_len];

        for (int ptr=0;ptr<array_len-1;ptr++){
            data_[ptr] = data[ptr] & 0xFF;
        }

        for (int i=0; i<256; i++){
            hist[i] = 0;
            cdf[i] = 0;
        }

        for (int j=0; j<array_len-1; j++){
            hist[(int)data_[j]] +=1;
        }

        int cdf_v = 0;
        int cdf_min = 0;
        int flag = 0;
        for (int k=0; k<256;k++){
            if(cdf_v != 0 && flag ==0){
                cdf_min = cdf_v;
                flag =1;
            }
            cdf_v += hist[k];
            cdf[k] += cdf_v;
        }

```

```
239
240             for (int l=0; l<256;l++){
241                 h[l] = ((cdf[l]-cdf_min)*255/(size-1));
242             }
243
244             int itr =0;
245             int temp;
246             for (int m=0; m<array_len-1; m++){
247                 itr = data_[m];
248                 temp = (h[itr]);
249                 data_o[m] = h[itr];
250                 if (temp > 255) temp = 255;
251                 if (temp<0) temp = 0;
252                 temp = (h[itr]&0xFF);
253
254                 histeqData[m] = (byte)temp;
255             }
256
257             // ********************** End YOUR CODE HERE  *************************** //
258             // We copy the colorness part for you, do not modify if you want rgb images
259             for(int i=size; i<data.length; i++){
260                 histeqData[i] = data[i];
261             }
262             return histeqData;
263         }
264
```

```
266     public byte get_pixel(byte[] data, int itr, int width, int height, double kernel[][]){
267         double return_val = 0;
268         double pic_data;
269         int off_x = (int)((kernel.length-1)/2);
270         int off_y = (int)((kernel[0].length-1)/2);
271         int x_cor;
272         int y_cor;
273 
274         for (int i=0; i<kernel.length;i++){
275             for (int j=0; j<kernel[i].length;j++){
276                 x_cor = itr%width - off_x +i;
277                 y_cor = (int)itr/width - off_y +j;
278                 if (x_cor<0 || x_cor >= width || y_cor <0 || y_cor >= height){
279                     pic_data = 0;
280                 }
281                 else{
282                     pic_data = (double)data[y_cor*width+x_cor];
283                 }
284                 return_val += pic_data*kernel[i][j];
285             }
286         }
287         return (byte)return_val;
288     }
289 
290     public int[] conv2(byte[] data, int width, int height, double kernel[][]){
291         // 0 is black and 255 is white.
292         int size = height * width;
293         int[] convData = new int[size];
294 
295         // Perform single channel 2D Convolution
296         // Note that you only need to manipulate data[0:size] that corresponds to luminance
297         // The rest data[size:data.length] is ignored since we only want grayscale output
298         // ********************** START YOUR CODE HERE  ************************** //
299         for (int itr = 0; itr<size;itr++){
300             convData[itr] = get_pixel(data,itr,width,height,kernel);
301         }
302         // ********************** End YOUR CODE HERE  ************************** //
303         return convData;
304     }
305 
306 }
```

Question:

When the original image has low dynamic range (the brightness gradient is low), the his_eq would be most significant.

For a 3x3 kernel on a 512x672 frame, we need 9 computation for each pixel making it 9*512*672 = 3096576