TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**TIỂU LUẬN GIỮA KÌ MÔN XÁC SUẤT VÀ THỐNG KÊ ỨNG DỤNG CHO CÔNG NGHỆ THÔNG TIN**

**STATISTICS LIBRARY IN PYTHON**

**HISTOGRAM EQUALIZATION**

*Người hướng dẫn*: **…**

*Người thực hiện*: **…**

Lớp **: …**

Khoá  **: …**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**TIỂU LUẬN GIỮA KÌ MÔN XÁC SUẤT VÀ THỐNG KÊ ỨNG DỤNG CHO CÔNG NGHỆ THÔNG TIN**

**STATISTICS LIBRARY IN PYTHON**

**HISTOGRAM EQUALIZATION**

Người hướng dẫn: **…**

Người thực hiện: **…**

Lớp **: …**

Khoá  **: …**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

# LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành đến giảng viên bộ môn – Thầy … đã truyền đạt những kiến thức quý báu cho em trong suốt thời gian học tập vừa qua. Trong thời gian tham gia lớp học Xác suất và thống kê ứng dụng cho Công nghệ thông tin của thầy, em đã có thêm cho mình nhiều kiến thức bổ ích qua đó thúc đẩy sự tò mò và hứng thú của em với môn học, ngành học hơn. Đây chắc chắn sẽ là những kiến thức quý báu, là hành trang để em có thể vững bước trong quá trình học tập của mình cũng như là khi đi làm.

Bài tiểu luận khó có thể tránh khỏi những thiếu sót và nhiều chỗ còn chưa chính xác, kính mong Thầy xem xét và góp ý để bài tiểu luận của em được hoàn thiện hơn.

Em xin chân thành cảm ơn.

*TP. Hồ Chí Minh, ngày … tháng … năm 2022*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*…*

**TIỂU LUẬN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Em xin cam đoan đây là bài tiểu luận của riêng em và được sự hướng dẫn của …. Các nội dung nghiên cứu trong các đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây.

Ngoài ra, trong tiểu luận còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào em xin hoàn toàn chịu trách nhiệm về nội dung tiểu luận của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do em gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày … tháng … năm 2022*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*…*

**PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN**

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

# MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc118833837)

[MỤC LỤC 4](#_Toc118833838)

[PHẦN 1 – STATISTICS LIBRARY IN PYTHON 6](#_Toc118833839)

[1.1 Tổng quan 6](#_Toc118833840)

[1.2 Tất cả các hàm trong thư viện Statistics 7](#_Toc118833841)

[*1.2.1 Trung bình và các phương pháp tìm giá trị đại diện* 7](#_Toc118833842)

[1.2.1.1 mean(data) 7](#_Toc118833843)

[1.2.1.2 fmean(data, weights) 8](#_Toc118833844)

[1.2.1.3 geometric\_mean(data) 9](#_Toc118833845)

[1.2.1.4 harmonic\_mean(data, weights) 10](#_Toc118833846)

[1.2.1.5 median(data) 11](#_Toc118833847)

[1.2.1.6 median\_low(data) 12](#_Toc118833848)

[1.2.1.7 median\_high(data) 14](#_Toc118833849)

[1.2.1.8 median\_grouped(data) 15](#_Toc118833850)

[1.2.1.9 mode(data) 15](#_Toc118833851)

[1.2.1.10 multimode(data) 17](#_Toc118833852)

[1.2.1.11 quantiles(data, \*, n=4, method='exclusive') 18](#_Toc118833853)

[*1.2.2 Đo lường độ phân tán* 19](#_Toc118833854)

[1.2.2.1 pstdev(data, mu) 19](#_Toc118833855)

[1.2.2.2 pvariance(data, mu) 20](#_Toc118833856)

[1.2.2.3 stdev(data, xbar) 20](#_Toc118833857)

[1.2.2.4 variance(data, xbar) 21](#_Toc118833858)

[*1.2.3 Thống kê quan hệ giữa 2 bộ dữ liệu* 22](#_Toc118833859)

[1.2.3.1 covariance(x, y, /) 22](#_Toc118833860)

[1.2.3.2 correlation(x, y, /) 23](#_Toc118833861)

[1.2.3.3 linear\_regression(x, y, /, \*, proportional=False) 24](#_Toc118833862)

[PHẦN 2 – HISTOGRAM EQUALIZATION ALGORITHM 26](#_Toc118833863)

[2.1 Tổng quan 26](#_Toc118833864)

[2.2 Ví dụ 31](#_Toc118833865)

[2.3 Nhận xét, phân tích, đánh giá 34](#_Toc118833866)

[PHẦN 3 – IMPLEMENTATION 35](#_Toc118833867)

# PHẦN 1 – STATISTICS LIBRARY IN PYTHON

## 1.1 Tổng quan

* Thư viện **Statistics** là thư viện được xây dựng sẵn trong ngôn ngữ Python phục vụ cho **thống kê mô tả** trên dữ liệu số.
* Thư viện cung cấp các chức năng để có được các số liệu thống kê của bộ dữ liệu số như trung bình, trung vị, yếu vị, độ lệch chuẩn, phương sai,…
* Ta có thể sử dụng thư viện này nếu các tập dữ liệu của ta không quá lớn hoặc ta không muốn sử dụng các thư viện của bên thứ ba

## 1.2 Tất cả các hàm trong thư viện Statistics

* Thư viện Statistics gồm 3 nhóm hàm chính:

### *1.2.1 Trung bình và các phương pháp tìm giá trị đại diện*

#### 1.2.1.1 mean(data)

* **Cách sử dụng:**
  + - Hàm **mean()** tính giá trị trung bình cộng của tập dữ liệu truyền vào
      * Giá trị trung bình cộng của tập dữ liệu bằng tổng các giá trị trong tập dữ liệu chia cho số lượng giá trị có trong tập dữ liệu
* **Tham số đầu vào:** 
  + - data (bắt buộc): 1 tập dữ liệu
      * Nếu tập dữ liệu rỗng thì trả về StatisticsError
* **Ý nghĩa của kết quả trả về:** 
  + - Giá trị trung bình cộng của tập dữ liệu truyền vào
* **Code ví dụ:**

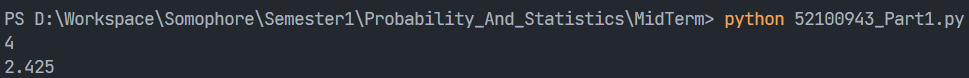
# Tính giá trị trung bình cộng của tập dữ liệu

print(mean([1, 3, 8]))

# (1 + 3 + 8) / 3 = 12 / 3 = 4

print(mean([3.2, 4.8, -1.2, 2.9]))

# (3.2 + 4.8 - 1.2 + 2.9) / 4 = 9.7 / 4 = 2.425



#### 1.2.1.2 fmean(data, weights)

* **Cách sử dụng:** 
  + - Hàm **fmean()** sẽ chuyển đổi các giá trị trong tập dữ liệu truyền vào thành kiểu dữ liệu float sau đó tính giá trị trung bình cộng của tập dữ liệu
      * Giá trị trung bình cộng của tập dữ liệu bằng tổng các giá trị trong tập dữ liệu chia cho số lượng giá trị có trong tập dữ liệu
* **Tham số đầu vào:** 
  + - data (bắt buộc): 1 tập dữ liệu
      * Nếu tập dữ liệu rỗng thì trả về StatisticsError
    - weights (tùy chọn):
      * Nếu weights được truyền vào thì số lượng trọng số phải bằng với số lượng giá trị trong tập dữ liệu nếu không sẽ trả về ValueError
* **Ý nghĩa của kết quả trả về:** 
  + - Giá trị trung bình cộng của tập dữ liệu truyền vào ở kiểu dữ liệu float
* **Code ví dụ:**

# Chuyển đổi các giá trị trong tập dữ liệu thành kiểu dữ liệu float

# Sau đó tính giá trị trung bình cộng của tập dữ liệu

print(fmean([1, 3, 8]))

# (1.0 + 3.0 + 8.0) / 3 = 12.0 / 3 = 4.0

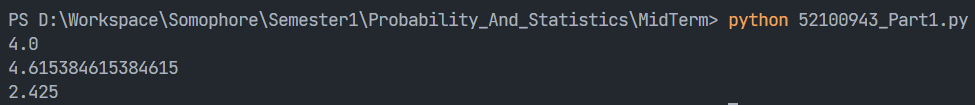
print(fmean([2, 4, 6], [0.2, 0.5, 0.6]))

# (2.0 \* 0.2 + 4.0 \* 0.5 + 6.0 \* 0.6) / (0.2 + 0.5 + 0.6)

# = 6.0 / 1.3

print(fmean([3.2, 4.8, -1.2, 2.9]))

# (3.2 + 4.8 - 1.2 + 2.9) / 4 = 9.7 / 4 = 2.425



#### 1.2.1.3 geometric\_mean(data)

* **Cách sử dụng:** 
  + - Hàm **geometric\_mean()** sẽ chuyển đổi các giá trị trong tập dữ liệu truyền vào thành kiểu dữ liệu float sau đó tính giá trị trung bình nhân của tập dữ liệu
      * Giá trị trung bình nhân của tập dữ liệu bằng căn bậc số lượng giá trị của tập dữ liệu của tích các giá trị trong tập dữ liệu
* **Tham số đầu vào:** 
  + - data (bắt buộc): 1 tập dữ liệu
      * Nếu tập dữ liệu rỗng hoặc có chứa giá trị 0 hoặc có chứa giá trị âm thì trả về sẽ StatisticsError
* **Ý nghĩa của kết quả trả về:**
  + - Giá trị trung bình nhân của tập dữ liệu truyền vào ở kiểu dữ liệu float
* **Code ví dụ:**

# Tính giá trị trung bình nhân của tập dữ liệu

print(geometric\_mean([32, 16, 40]))

# Căn bậc 3 của (32 \* 16 \* 40) = (32 \* 16 \* 40) \*\* (1/3)

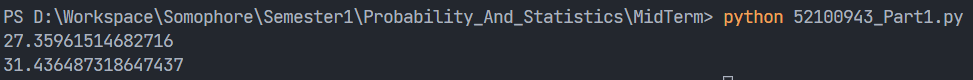
# Vì tập dữ liệu có 3 giá trị nên ta lấy căn bậc 3

print(geometric\_mean([24, 15, 9, 92, 103]))

# Căn bậc 5 của (24 \* 15 \* 9 \* 92 \* 103) = (24 \* 15 \* 9 \* 92 \* 103) \*\* (1/5)

# Vì tập dữ liệu có 5 giá trị nên ta lấy căn bậc 5

# => Tập dữ liệu có n giá trị ta lấy căn bậc n của tích n giá trị



#### 1.2.1.4 harmonic\_mean(data, weights)

* **Cách sử dụng:**
  + - Hàm **harmonic\_mean()** tính giá trị trung bình điều hòa của tập dữ liệu truyền vào
      * Giá trị trung bình điều hòa của tập dữ liệu
        + : trọng số thứ i tương ứng với giá trị thứ i trong tập dữ liệu
        + : giá trị thứ i trong tập dữ liệu
* **Tham số đầu vào:**
  + - data (bắt buộc): 1 tập dữ liệu
      * Nếu tập dữ liệu rỗng hoặc có chứa giá trị < 0 thì trả về StatisticsError
    - weights (tùy chọn):
      * Nếu không truyền weights hoặc weights = None thì các trọng số tương ứng với các giá trị trong tập dữ liệu sẽ đều bằng nhau và bằng 1
      * Nếu truyền weights thì số lượng trọng số phải bằng với số lượng giá trị có trong tập dữ liệu
      * Nếu trong weights có trọng số < 0 thì trả về StatisticsError
* **Ý nghĩa của kết quả trả về:**
  + - Giá trị trung bình điều hòa của tập dữ liệu truyền vào
* **Code ví dụ:**

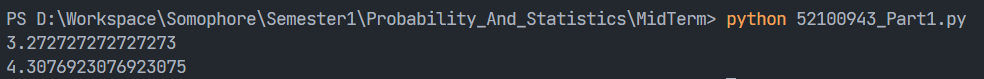
# Tính giá trị trung bình điều hòa của tập dữ liệu

print(harmonic\_mean([2, 4, 6]))

# (1 + 1 + 1) / (1/2 + 1/4 + 1/6) = 3 / (1/2 + 1/4 + 1/6) = 3 / (11/12) = 36 / 11

print(harmonic\_mean([2, 4, 6], [2, 3, 9]))

# (2 + 3 + 9) / (2/2 + 3/4 + 9/6) = 14 / (13/4) = 56 / 13



#### 1.2.1.5 median(data)

* **Cách sử dụng:** 
  + - Hàm **median()** tính trung vị của tập dữ liệu truyền vào
      * Đầu tiên hàm sẽ sắp xếp các giá trị trong bộ dữ liệu theo thứ tự tăng dần
      * Khi số lượng giá trị có trong tập dữ liệu là số lẻ thì trả về giá trị ở giữa tập dữ liệu
      * Khi số lượng giá trị có trong tập dữ liệu là số chẵn thì trả về giá trị trung bình của 2 trung vị của tập dữ liệu
* **Tham số đầu vào:** 
  + - data (bắt buộc): 1 tập dữ liệu
      * Nếu tập dữ liệu rỗng thì trả về StatisticsError
* **Ý nghĩa của kết quả trả về:**
  + - Trung vị của tập dữ liệu truyền vào
* **Code ví dụ:**

# Tìm trung vị của tập dữ liệu

print(median([1, 11, 13, 5, 7, 10]))

# Đầu tiên tập dữ liệu sẽ được sắp xếp theo thứ tự tăng dần

# [1, 5, 7, 10, 11, 13]

# Số lượng giá trị của tập dữ liệu là 6 (chẵn)

# => Trung vị = (7 + 10) / 2 = 8.5

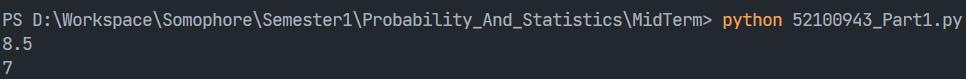
print(median([1, 11, 13, 5, 7]))

# Đầu tiên tập dữ liệu sẽ được sắp xếp theo thứ tự tăng dần

# [1, 5, 7, 11, 13]

# Số lượng giá trị của tập dữ liệu là 5 (lẻ)

# => Trung vị = 7



#### 1.2.1.6 median\_low(data)

* **Cách sử dụng:** 
  + - Hàm **median\_low()** trả về trung vị nhỏ hơn của tập dữ liệu truyền vào
      * Đầu tiên hàm sẽ sắp xếp các giá trị trong tập dữ liệu theo thứ tự tăng dần
      * Khi số lượng giá trị có trong tập dữ liệu là số lẻ thì trả về giá trị ở giữa tập dữ liệu
      * Khi số lượng giá trị có trong tập dữ liệu là số chẵn thì trả về trung vị nhỏ hơn trong 2 trung vị của tập dữ liệu
* **Tham số đầu vào:** 
  + - data (bắt buộc): 1 tập dữ liệu
      * Nếu tập dữ liệu rỗng thì trả về StatisticsError
* **Ý nghĩa của kết quả trả về:**
  + - Trung vị nhỏ hơn của tập dữ liệu truyền vào
* **Code ví dụ:**

# Tìm trung vị nhỏ hơn của tập dữ liệu

print(median\_low([1, 11, 13, 5, 7, 9]))

# Đầu tiên tập dữ liệu sẽ được sắp xếp theo thứ tự tăng dần

# [1, 5, 7, 9, 11, 13]

# Số lượng giá trị của tập dữ liệu là 6 (chẵn)

# => Trung vị nhỏ hơn = 7

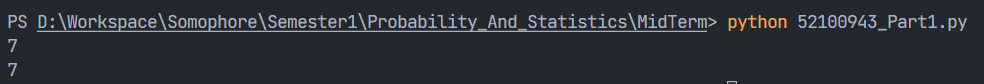
print(median\_low([1, 11, 13, 5, 7]))

# Đầu tiên tập dữ liệu sẽ được sắp xếp theo thứ tự tăng dần

# [1, 5, 7, 11, 13]

# Số lượng giá trị của tập dữ liệu là 5 (lẻ)

# => Trung vị = 7



#### 1.2.1.7 median\_high(data)

* **Cách sử dụng:** 
  + - Hàm **median\_high()** tìm trung vị lớn hơn của tập dữ liệu truyền vào
      * Đầu tiên hàm sẽ sắp xếp các giá trị trong bộ dữ liệu theo thứ tự tăng dần
      * Khi số lượng giá trị có trong tập dữ liệu là số lẻ thì trả về giá trị ở giữa tập dữ liệu
      * Khi số lượng giá trị có trong tập dữ liệu là số chẵn thì trả về trung vị lớn hơn trong 2 trung vị của tập dữ liệu
* **Tham số đầu vào:** 
  + - data (bắt buộc): 1 tập dữ liệu
      * Nếu tập dữ liệu rỗng thì trả về StatisticsError
* **Ý nghĩa của kết quả trả về:**
  + - Trung vị lớn hơn của tập dữ liệu
* **Code ví dụ:**

# Tìm trung vị lớn hơn của tập dữ liệu

print(median\_high([1, 11, 13, 5, 7, 9]))

# Đầu tiên tập dữ liệu sẽ được sắp xếp theo thứ tự tăng dần

# [1, 5, 7, 9, 11, 13]

# Số lượng giá trị của tập dữ liệu là 6 (chẵn)

# => Trung vị lớn hơn = 9

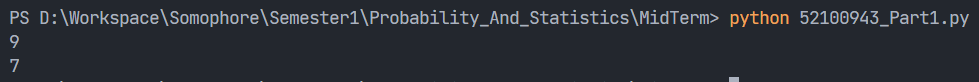
print(median\_high([1, 11, 13, 5, 7]))

# Đầu tiên tập dữ liệu sẽ được sắp xếp theo thứ tự tăng dần

# [1, 5, 7, 11, 13]

# Số lượng giá trị của tập dữ liệu là 5 (lẻ)

# => Trung vị = 7

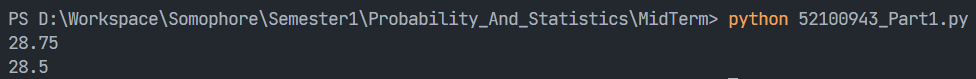


#### 1.2.1.8 median\_grouped(data)

* **Cách sử dụng**:
  + - Hàm **median\_grouped()** tính trung vị của dữ liệu liên tục được nhóm lại, được tính là phần trăm thứ 50
* **Tham số đầu vào**:
  + - data (bắt buộc): 1 tập dữ liệu
      * Nếu tập dữ liệu rỗng thì trả về StatisticsError
    - interval (tùy chọn): mặc đinh là 1
* **Ý nghĩa của kết quả trả về**:
  + - Trung vị của dữ liệu liên tục được nhóm lại, được tính là phần trăm thứ 50
* **Code ví dụ**:

print(median\_grouped([23, 28, 29, 29, 30]))

print(median\_grouped([23, 28, 29, 29, 30], 2))



#### 1.2.1.9 mode(data)

* **Cách sử dụng:** 
  + - Hàm **mode()** tìm yếu vị của tập dữ liệu truyền vào
      * Yếu vị: giá trị xuất hiện nhiều lần nhất trong tập dữ liệu
      * Nếu có nhiều yếu vị với tần số xuất hiện như nhau thì trả về yếu vị đầu tiên trong tập dữ liệu
      * Có thể áp dụng được cho tập dữ liệu định tính
* **Tham số đầu vào:**
  + - data (bắt buộc): 1 tập dữ liệu
      * Nếu tập dữ liệu rỗng trong tập dữ liệu thì trả về StatisticsError
* **Ý nghĩa của kết quả trả về:**
  + - Yếu vị của tập dữ liệu
* **Code ví dụ:**

# Tìm yếu vị của tập dữ liệu

print(mode([25, 26, 27, 27, 28, 29, 28, 28, 30]))

# Trong tập dữ liệu, giá trị 28 xuất hiện nhiều nhất với số lần xuất hiện là 3

# => Yếu vị = 28

print(mode([25, 26, 27, 27, 27, 28, 29, 28, 28, 30]))

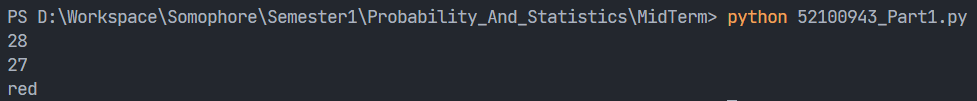
# Trong tập dữ liệu, giá trị 27 và 28 xuất hiện nhiều nhất với số lần xuất hiện là 3

# Tuy nhiên chỉ trả về yếu vị đầu tiên trong tập dữ liệu (trong trường hợp này là 27)

print(mode(["red", "blue", "blue", "red", "green", "red", "red"]))

# Trong tập dữ liệu, "red" xuất hiện nhiều nhất với số lần xuất hiện là 3

# Yếu vị là "red"



#### 1.2.1.10 multimode(data)

* **Cách sử dụng:** 
  + - Hàm **multimode()** trả về danh sách các yếu vị trong tập dữ liệu
* **Tham số đầu vào:**
  + - data (bắt buộc): 1 tập dữ liệu
      * Nếu tập dữ liệu rỗng thì trả về danh sách rỗng
* **Ý nghĩa của kết quả trả về:**
  + - Danh sách các yếu vị của tập dữ liệu
* **Code ví dụ:**

# Tìm danh sách các yếu vị của tập dữ liệu

print(multimode([25, 26, 27, 27, 27, 28, 29, 28, 28, 29, 29, 30]))

# Trong tập dữ liệu, giá trị 27, 28 và 29 xuất hiện nhiều nhất với số lần xuất hiện là 3

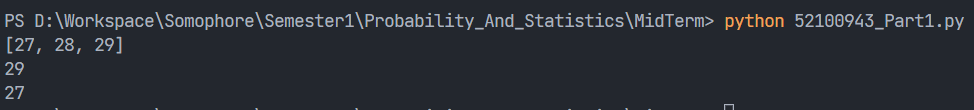
# Kết quả trả về là 27, 28, 29

# Tìm yếu vị lớn nhất: 29

print(max(multimode([25, 26, 27, 27, 27, 28, 29, 28, 28, 29, 29, 30])))

# Tìm yếu vị nhỏ nhất: 27

print(min(multimode([25, 26, 27, 27, 27, 28, 29, 28, 28, 29, 29, 30])))



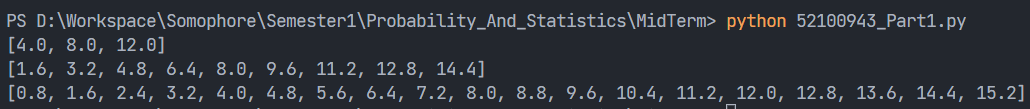
#### 1.2.1.11 quantiles(data, \*, n=4, method='exclusive')

* **Cách sử dụng:** 
  + - Hàm **quantiles()** chia tập dữ liệu thành n khoảng liên tục với xác suất bằng nhau. Trả về danh sách của n - 1 điểm cắt ngăn cách các khoảng
* **Tham số đầu vào:**
  + - data (bắt buộc): 1 tập dữ liệu
      * Nếu tập dữ liệu có ít hơn 2 giá trị thì trả về StatisticsError
    - n (tùy chọn): mặc định là 4
    - method
* **Ý nghĩa của kết quả trả về:**
  + - Danh sách của n - 1 điểm cắt ngăn cách các khoảng
* **Code ví dụ:**

print(quantiles([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15], n = 4))

print(quantiles([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15], n = 10))

print(quantiles([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15], n = 20))



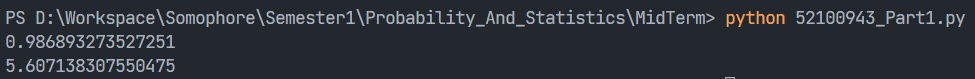
### *1.2.2 Đo lường độ phân tán*

#### 1.2.2.1 pstdev(data, mu)

* **Cách sử dụng:**
  + - Hàm **pstdev()** tính độ lệch chuẩn của tập dữ liệu (**tổng thể**)
      * Độ lệch chuẩn của tập dữ liệu (**tổng thể**) bằng căn bậc hai của phương sai của tập dữ liệu (**tổng thể**)
* **Tham số đầu vào:** 
  + - data (bắt buộc): 1 tập dữ liệu (**tổng thể**)
      * Nếu tập dữ liệu rỗng thì trả về StatisticsError
    - mu (tùy chọn): giá trị trung bình của tập dữ liệu truyền vào
      * Nếu không truyền mu hoặc mu = None thì giá trị trung bình của tập dữ liệu truyền vào sẽ được tự động tính toán
* **Ý nghĩa của kết quả trả về:**
  + - Độ lệch chuẩn của tập dữ liệu (**tổng thể**)
* **Code ví dụ:**

print(pstdev([1.5, 2.5, 2.5, 2.75, 3.25, 4.75]))

print(pstdev([25, 29, 31, 38, 40]))



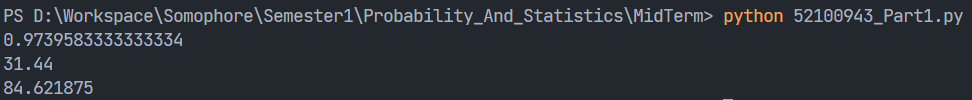
#### 1.2.2.2 pvariance(data, mu)

* **Cách sử dụng:** 
  + - Hàm **pvariance()** tính phương sai của tập dữ liệu (**tổng thể**)
* **Tham số đầu vào:** 
  + - data (bắt buộc): 1 tập dữ liệu (**tổng thể**)
      * Nếu tập dữ liệu rỗng thì trả về StatisticsError
    - mu (tùy chọn): giá trị trung bình của tập dữ liệu truyền vào
      * Nếu không truyền mu hoặc mu = None thì giá trị trung bình của tập dữ liệu truyền vào sẽ được tự động tính toán
* **Ý nghĩa của kết quả trả về:**
  + - Phương sai của tập dữ liệu (**tổng thể**)
* **Code ví dụ**:

print(pvariance([1.5, 2.5, 2.5, 2.75, 3.25, 4.75]))

print(pvariance([25, 29, 31, 38, 40]))

print(pvariance([5, 18.6, 4.2, 25.9], mean([5, 18.6, 4.2, 25.9])))

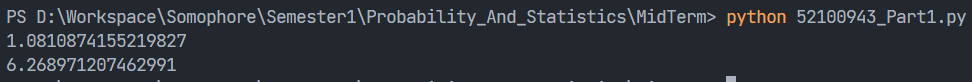


#### 1.2.2.3 stdev(data, xbar)

* **Cách sử dụng:**
  + - Hàm **stdev()** tính độ lệch chuẩn của tập dữ liệu (**mẫu**)
      * Độ lệch chuẩn của tập dữ liệu (**mẫu**) bằng căn bậc hai của phương sai của tập dữ liệu (**mẫu**)
* **Tham số đầu vào:** 
  + - data (bắt buộc): 1 tập dữ liệu (**mẫu**)
      * Nếu tập dữ liệu có ít hơn 2 giá trị thì trả về StatisticsError
    - mu (tùy chọn): giá trị trung bình của tập dữ liệu truyền vào
      * Nếu không truyền mu hoặc mu = None thì giá trị trung bình của tập dữ liệu truyền vào sẽ được tự động tính toán
* **Ý nghĩa của kết quả trả về:**
  + - Độ lệch chuẩn của tập dữ liệu (**mẫu**)
* **Code ví dụ:**

print(stdev([1.5, 2.5, 2.5, 2.75, 3.25, 4.75]))

print(stdev([25, 29, 31, 38, 40]))



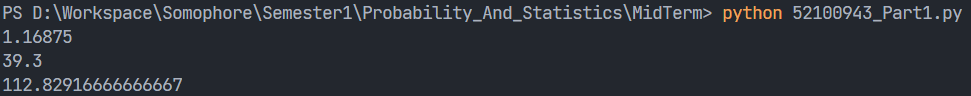
#### 1.2.2.4 variance(data, xbar)

* **Cách sử dụng:** 
  + - Hàm **variance()** tính phương sai của tập dữ liệu (**mẫu**)
* **Tham số đầu vào:** 
  + - data (bắt buộc): 1 tập dữ liệu (**mẫu**)
      * Nếu tập dữ liệu có ít hơn 2 giá trị thì trả về StatisticsError
    - mu (tùy chọn): giá trị trung bình của tập dữ liệu truyền vào
      * Nếu không truyền mu hoặc mu = None thì giá trị trung bình của tập dữ liệu truyền vào sẽ được tự động tính toán
* **Ý nghĩa của kết quả trả về:**
  + - Phương sai của tập dữ liệu (**mẫu**)
* **Code ví dụ**:

print(variance([1.5, 2.5, 2.5, 2.75, 3.25, 4.75]))

print(variance([25, 29, 31, 38, 40]))

print(variance([5, 18.6, 4.2, 25.9], mean([5, 18.6, 4.2, 25.9])))



### *1.2.3 Thống kê quan hệ giữa 2 bộ dữ liệu*

#### 1.2.3.1 covariance(x, y, /)

* **Cách sử dụng**:
  + - Hàm **covariance()** tính hiệp phương sai của 2 tập dữ liệu truyền vào
* **Tham số đầu vào**:
  + - x: tập dữ liệu thứ nhất
    - y: tập dữ liệu thứ hai
      * Cả 2 tập dữ liệu truyền vào phải có cùng kích thước và không có ít hơn 2 giá trị nếu không sẽ trả về StatisticsError
* **Ý nghĩa của kết quả trả về**:
  + - Hiệp phương sai của 2 tập dữ liệu truyền vào
* **Code ví dụ**:

x = [1, 3, 5, 9, 8, 7, 6]

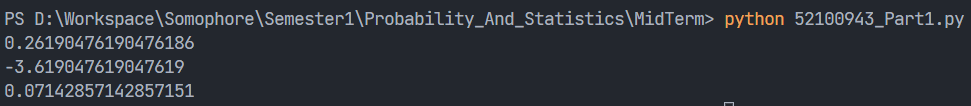
y = [1, 2, 3, 1, 2, 3, 1]

z = [8, 5, 3, 1, 8, 4, 2]

print(covariance(x, y))

print(covariance(x, z))

print(covariance(y, z))



#### 1.2.3.2 correlation(x, y, /)

* **Cách sử dụng**:
  + - Hàm correlation() trả về **hệ số tương quan Pearson của 2 tập dữ liệu truyền vào**
      * **Hệ số tương quan Pearson** (r) có giá trị giao động trong khoảng liên tục từ -1 đến +1
        + r = 0: Hai tập dữ liệu không có tương quan tuyến tính
        + r = 1; r = -1: Hai tập dữ liệu có mối tương quan tuyến tính tuyệt đối
* **Tham số đầu vào**:
  + - x: tập dữ liệu thứ nhất
    - y: tập dữ liệu thứ hai
      * Cả 2 tập dữ liệu truyền vào phải có cùng kích thước và không có ít hơn 2 giá trị nếu không sẽ trả về StatisticsError
* **Ý nghĩa của kết quả trả về**:
  + - Hệ **số tương quan Pearson của 2 tập dữ liệu truyền vào**
* **Code ví dụ**:

# Ví dụ ta muốn biết hệ số tương quan giữa nhiệt độ trong ngày với doanh thu bán kem

temp = [35, 30, 25] # nhiệt độ trung bình mỗi ngày trong 3 ngày gần đây, đơn vị độ C

revenue = [800, 600, 250] # doanh thu bán kem mỗi ngày trong 3 ngày gần đây, đơn vị $

print(correlation(temp, revenue))

# Hệ số tương quan dương có nghĩa là nhiệt độ trung bình trong ngày cao thì lượng người mua kem trong ngày cũng sẽ cao dẫn đến doanh thu bán kem cao vào những ngày này



#### 1.2.3.3 linear\_regression(x, y, /, \*, proportional=False)

* **Cách sử dụng**:
  + - Hàm **linear\_regression**() trả về slope và intercept của các tham số hồi quy tuyến tính đơn giản được ước tính bằng cách sử dụng bình phương tối thiểu thông thường
* **Tham số đầu vào**:
  + - x: tập dữ liệu thứ nhất
    - y: tập dữ liệu thứ hai
      * Cả 2 tập dữ liệu truyền vào phải có cùng kích thước và không có ít hơn 2 giá trị nếu không sẽ trả về StatisticsError
* **Ý nghĩa của kết quả trả về:**
  + - Slope và intercept của các tham số hồi quy tuyến tính đơn giản được ước tính bằng cách sử dụng bình phương tối thiểu thông thường
* **Code ví dụ**:

# Ví dụ ta muốn dự đoán chiều cao của một người dựa vào số tuổi

age = [1, 5, 13, 17, 24]

height = [50, 100, 140, 169, 180]

slope, intercept = linear\_regression(age, height)

print(round(slope \* 18 + intercept)) # Ta muốn dự đoán một người 18 tuổi có chiều cao bao nhiêu



# PHẦN 2 – HISTOGRAM EQUALIZATION ALGORITHM

## 2.1 Tổng quan

* Vấn đề:

Trong cuộc sống, ta có thể dễ dàng bắt gặp được các bức ảnh có những đặc điểm như sau:

* Ảnh quá sáng



* Ảnh quá tối



Vậy làm sao để ta có thể xử lý những bức ảnh này để có được kết quả như sau:



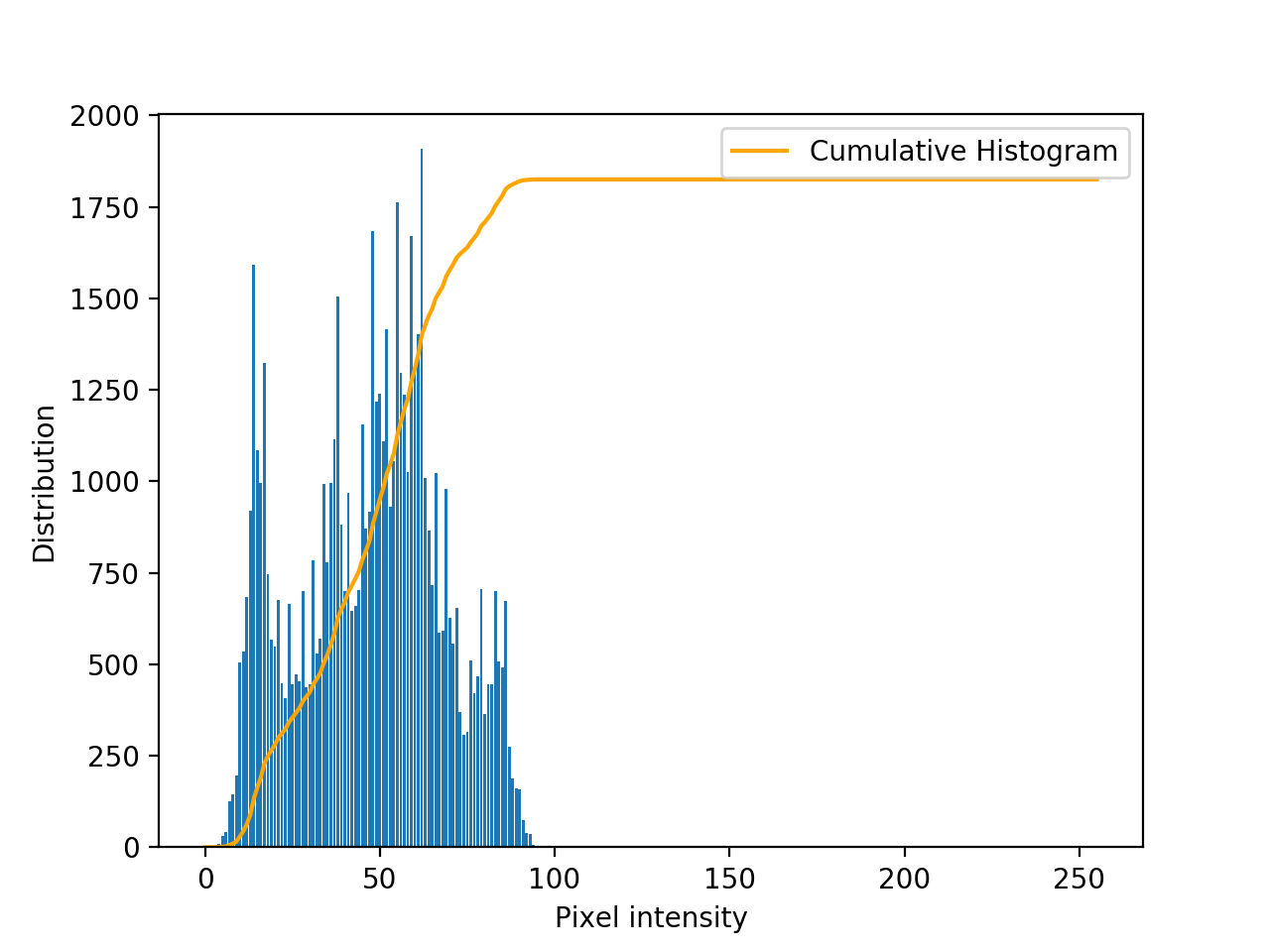


Chính nhờ đến thuật toán Histogram equalization mà ta có thể giải quyết được vấn đề này.

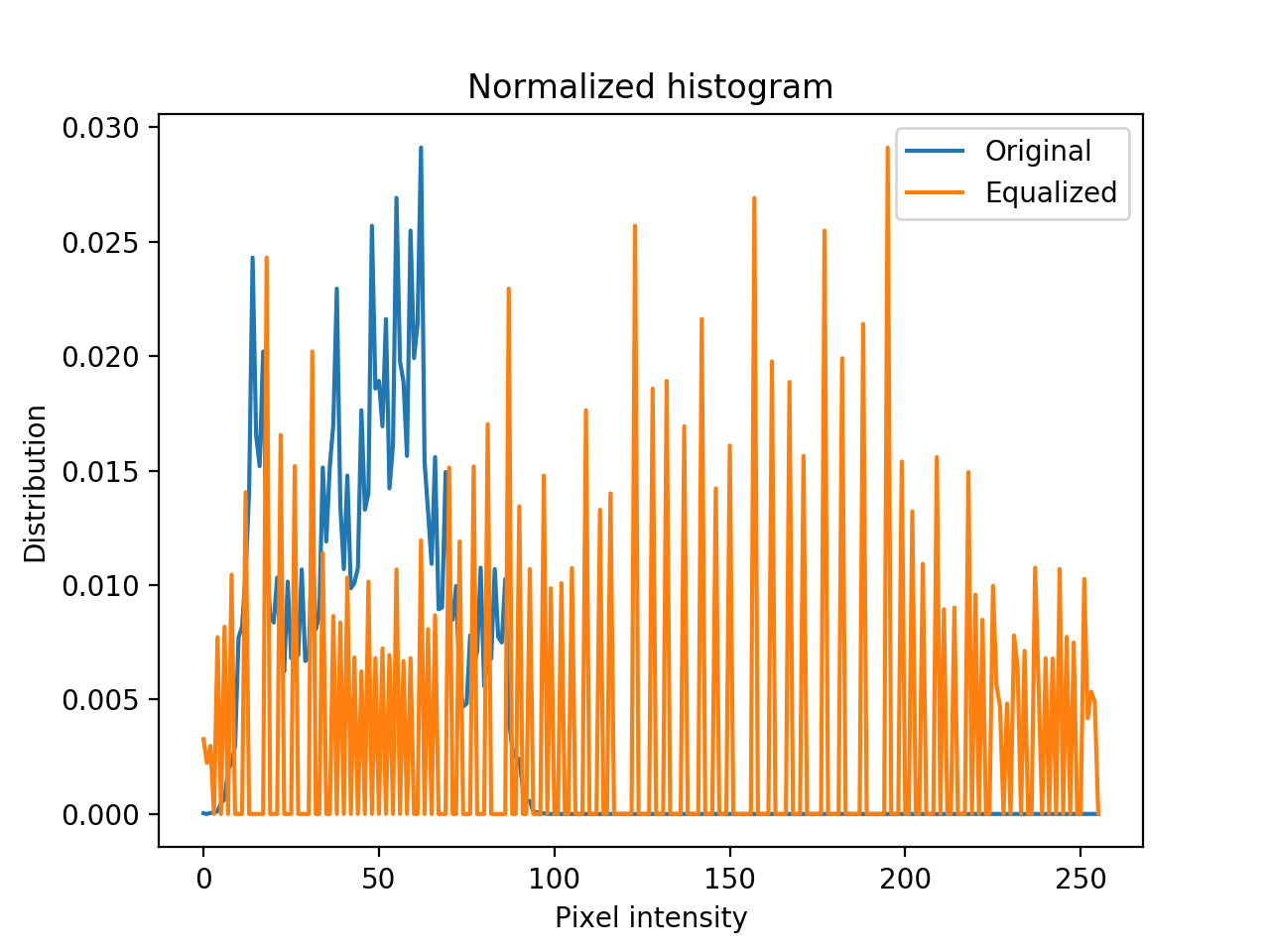
* Histogram của ảnh
* Là biểu đồ phân bố cường độ sáng của ảnh. Trong Histogram của ảnh, trục hoành biểu diễn các giá trị của cường độ sáng (thường là số nguyên từ 0 đến 255) và trục tung biểu diễn cho số lượng pixel có giá trị cường độ sáng ứng với trục hoành.



* Ví dụ đối với ảnh trên, ta có histogram của ảnh như sau:



* Mục tiêu
  + Tăng độ tương phản cho ảnh bằng cách hiệu chỉnh lại histogram của ảnh, cụ thể là phân bố cường độ sáng trải đều từ giá trị 0 đến 255
  + Ví dụ như histogram của ảnh trên, ta thấy cường độ sáng chỉ tập trung ở khoảng từ 0 đến 100 để tăng độ tương phản cho ảnh ta cần phải phân bố cường độ sáng trải đều từ giá trị 0 đến 255 như sau.



Sau khi hiệu chỉnh lại histogram của ảnh bằng cách phân bố cường độ sáng trải đều từ giá trị 0 đến 255, ta sẽ thu được ảnh có kết quả như sau



* Ràng buộc/Điều kiện:
  + - Thuật toán Histogram equalization xử lý hiệu quả nhất trên ảnh xám (grayscale - thước xám với 256 cấp độ xám từ 0 đến 255)
* Thuật toán:
  + - Bước 1: Tính tần suất tích lũy hay còn gọi là mật độ xác suất xuất hiện của mỗi mức xám có trong ảnh
      * Trong đó:
        + n: tổng số pixel của ảnh
        + hoặc h(: số lượng pixel của mức xám thứ k
        + : giá trị mức xám thứ k với k = 0 đến L – 1

L: số mức xám trong ảnh

* Bước 2: Biến đổi mức xám thành mức xám qua phép biến đổi T
* Bước 3: Ánh xạ các mức xám biến đổi được cho số lượng pixel tương ứng

## 2.2 Ví dụ

* Giả sử ta có một ảnh 3 bit **⟹** L = 8, có kích thước 64 x 64 **⟹** n = 4096, với các pixel được phân bố ở các mức xám như bảng dưới:

|  |  |  |  |
| --- | --- | --- | --- |
| k |  |  |  |
| 0 | 0 | 81 | 0.02 |
| 1 | 1 | 122 | 0.03 |
| 2 | 2 | 245 | 0.06 |
| 3 | 3 | 329 | 0.08 |
| 4 | 4 | 656 | 0.16 |
| 5 | 5 | 850 | 0.21 |
| 6 | 6 | 1023 | 0.25 |
| 7 | 7 | 790 | 0.19 |

* Biến đổi mức xám thành mức xám qua phép biến đổi T
  + Với k = 0:
  + Với k = 1:
  + Với k = 2:
  + Với k = 3:
  + Với k = 4:
  + Với k = 5:
  + Với k = 6:
  + Với k = 7:
* Ánh xạ các mức xám biến đổi được cho số lượng pixel tương ứng
  + Với 81 pixel có mức xám ⟹ 81 pixel này sẽ ánh xạ qua mức xám
  + Với 122 pixel có mức xám ⟹ 122 pixel này sẽ ánh xạ qua mức xám
  + Với 245 pixel có mức xám ⟹ 245 pixel này sẽ ánh xạ qua mức xám
  + Với 329 pixel có mức xám ⟹ 329 pixel này sẽ ánh xạ qua mức xám
  + Với 656 pixel có mức xám ⟹ 656 pixel này sẽ ánh xạ qua mức xám
  + Với 850 pixel có mức xám ⟹ 850 pixel này sẽ ánh xạ qua mức xám
  + Với 1023 pixel có mức xám ⟹ 1023 pixel này sẽ ánh xạ qua mức xám
  + Với 790 pixel có mức xám ⟹ 790 pixel này sẽ ánh xạ qua mức xám

|  |  |  |  |
| --- | --- | --- | --- |
| k |  |  |  |
| 0, 1 | 0 | 81 + 122 = 203 | 0.05 |
| 2, 3 | 1 | 245 + 329 = 574 | 0.14 |
| 4 | 2 | 656 | 0.16 |
| 5 | 4 | 850 | 0.21 |
| 6 | 6 | 1023 | 0.25 |
| 7 | 7 | 790 | 0.19 |

## 2.3 Nhận xét, phân tích, đánh giá

* Ứng dụng:
  + - Áp dụng rộng rãi vào xử lý ảnh trong y học như tăng độ tương phản của X-ray, MRI khiến cho việc chẩn đoán chính xác hơn
    - Các giải thuật xử lý ảnh thường ảnh hưởng bởi độ sáng, cùng một nội dung ảnh nhưng với các điều kiện ánh sáng khác nhau có thể làm sai lệch kết quả xử lý. Do đó Histogram equalization là một trong những cách giúp làm giảm các ảnh hưởng này

# PHẦN 3 – IMPLEMENTATION

* Đầu tiên, ta cần import các thư viện cần thiết hỗ trợ cho việc cài đặt thuật toán:
  + Module numpy hỗ trợ trong việc thao tác với mảng, ma trận,…
  + Module pyplot trong module matplotlib hỗ trợ vẽ biểu đồ, đồ thị,...
  + Module Image trong module PIL hỗ trợ thao tác với ảnh

# Import các thư viện cần thiết

import numpy as np

from matplotlib import pyplot as plt

from PIL import Image

* Tiếp theo, ta mở ảnh muốn xử lý bằng cách truyền đường dẫn của ảnh vào

# Mở ảnh

img = Image.open('assets/dark\_image.jpg')



* Trong trường hợp ảnh ta truyền vào là ảnh màu như RGB,… thì ta biến đổi sang ảnh xám vì thuật toán Histogram equalization chỉ xử lý hiệu quả trên ảnh xám

# Biến đổi ảnh màu thành ảnh xám

img = img.convert('L')

* Tính số lượng pixels có trong ảnh

# Số lượng pixels trong ảnh = chiều rộng ảnh x chiều cao ảnh

numberOfPixels = img.size[0] \* img.size[1]

* Đối với bất kì bài toán xử lý ảnh nào, ta đều cần biến đổi ảnh sang ma trận số để xử lý:

# Biến đổi ảnh thành ma trận 2 chiều với

# số dòng là chiều cao ảnh, số cột là chiều rộng ảnh

matrix\_img = np.asarray(img)

* Để thuận tiện cho việc đếm số lượng xuất hiện của các giá trị mức xám, ta sẽ biến đổi ma trận thành mảng một chiều

# Biến đổi ma trận thành mảng 1 chiều => Thuận tiện cho việc đếm số lần xuất hiện của các giá trị mức xám

pixels = matrix\_img.flatten()

* Đếm số lần xuất hiện của các giá trị mức xám và lưu vào mảng

# Biểu đồ phân bố tần số của các giá trị mức xám

def getImageHistogram(matrix\_img):

    # Tạo mảng với kích thước là 256 để lưu số lần xuất hiện của các giá trị mức xám từ 0 đến 255

    # Giá trị mặc định của mỗi phần tử trong mảng là 0

    histogram = np.zeros(256)

    # Các pixels trong ảnh chỉ có giá trị mức xám từ 0 đến 255

    # => Các giá trị mức xám cũng chính là indexes của mảng

    # Nếu xuất hiện giá trị mức xám nào thì ta sẽ cộng thêm 1 lần xuất hiện cho phần tử lưu số lần xuất hiện của giá trị mức xám tương ứng

    for pixel in matrix\_img:

        histogram[pixel] += 1

    return histogram

imgHist = getImageHistogram(pixels)

* Để thấy sự phân bố của các mức xám trước khi áp dụng thuật toán Histogram equalization thì ta sẽ vẽ đồ thị phân bố tần số của các giá trị mức xám trong ảnh

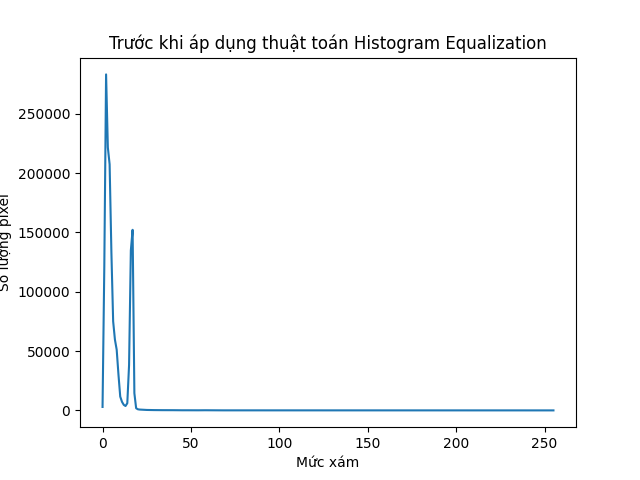
plt.plot(imgHist)

plt.title('Trước khi áp dụng thuật toán Histogram Equalization')

plt.xlabel('Mức xám')

plt.ylabel('Số lượng pixel')

plt.show()



* + Như ta thấy, các mức xám chỉ tập trung ở khoảng từ 0 đến 25
* Tính tần suất tích lũy hay còn gọi là mật độ xác suất xuất hiện của mỗi giá trị mức xám có trong ảnh

# Tính tần suất tích lũy

# f: mảng chứa tần số của các giá trị mức xám

def cumulativeRelativeFrequencyDistributions(f, N):

    # N: Số lượng pixels trong ảnh

    result = [f[0] / N]  # tạo mảng để lưu trữ tần suất tích lũy

    for i in range(1, len(f)):

        result.append(result[-1] + (f[i] / N))

    return np.array(result)

crfd = cumulativeRelativeFrequencyDistributions(imgHist, numberOfPixels)

* Biến đổi mức xám thành mức xám qua phép biến đổi T

def equalHist(crfd):

    for i in range(len(crfd)):

        crfd[i] \*= 255

        crfd[i] = int(crfd[i])

    return crfd

sk = equalHist(crfd)

* Ánh xạ các mức xám biến đổi được cho số lượng pixel tương ứng

# Ánh xạ các mức xám biến đổi được cho số lượng pixel tương ứng

pixels\_new = sk[pixels]

* Để thấy sự phân bố của các mức xám sau khi áp dụng thuật toán Histogram equalization thì ta sẽ vẽ biểu đồ phân bố tần số của các giá trị mức xám trong ảnh

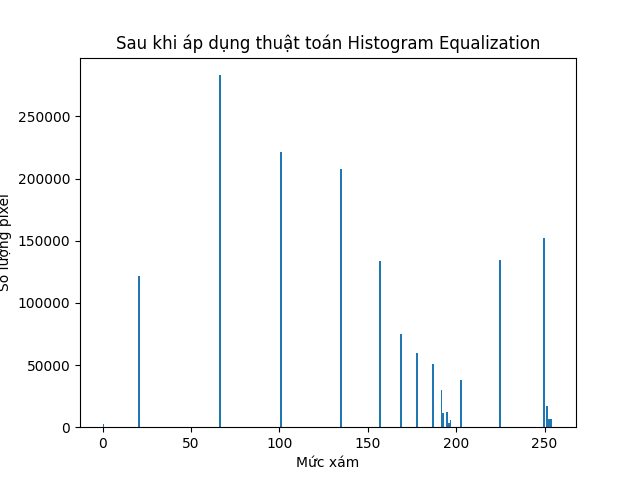
plt.hist(pixels\_new, bins=256)

plt.title('Sau khi áp dụng thuật toán Histogram Equalization')

plt.xlabel('Mức xám')

plt.ylabel('Số lượng pixel')

plt.show()



* + Như ta thấy, các mức xám đã phân bố đều ở các giá trị từ 0 đến 255.
* Biến đổi mảng một chiều thành ma trận

# Biến mảng trở lại thành ma trận vì lúc đầu ta biến ma trận thành mảng 1 chiều để thuận tiện đếm số lần xuất hiện của các giá trị xám

matrix\_img\_new = np.reshape(pixels\_new, matrix\_img.shape)

* Hiển thị kết quả

# Hiển thị ảnh trước và sau khi áp dụng thuật toán

fig = plt.figure()

fig.set\_figheight(15)

fig.set\_figwidth(15)

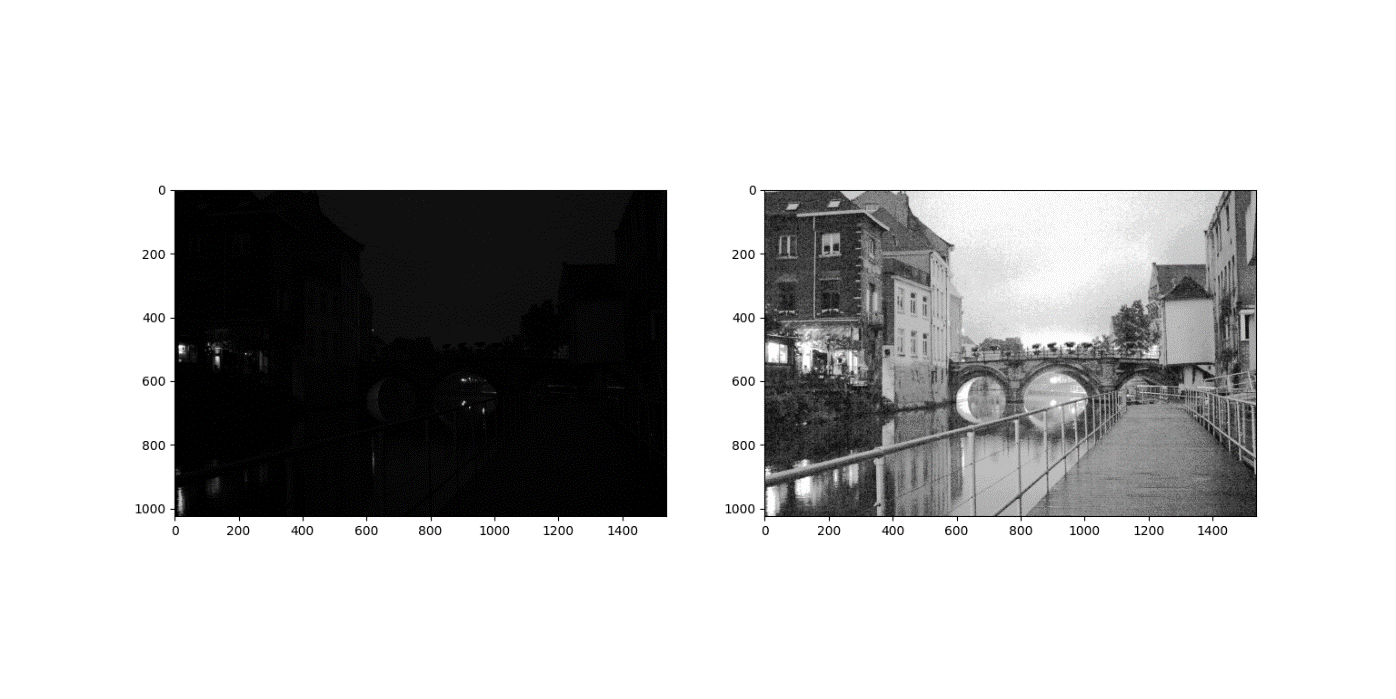
fig.add\_subplot(1, 2, 1)

plt.imshow(matrix\_img, cmap='gray')

fig.add\_subplot(1, 2, 2)

plt.imshow(matrix\_img\_new, cmap='gray')

plt.show(block=True)



* + Ảnh bên trái là ảnh đầu vào
  + Ảnh bên phải là ảnh sau khi áp dụng thuật toán Histogram equalization

**TÀI LIỆU THAM KHẢO**

**Tiếng Việt**

[1]. [Xử lý ảnh với OpenCV - Tut5: Histogram Equalization | GMO-Z.com Vietnam Lab Center](https://blog.vietnamlab.vn/xu-ly-anh-voi-opencv-tut5-can-bang-histogram/)

[2]. [Tăng cường ảnh - Xử lý ảnh - TS. Nguyễn Thanh Hải - YouTube](https://www.youtube.com/watch?v=KbLEPEYHvdU)

[3]. [Xử lý ảnh - OpenCV cân bằng sáng (histogram equalization) (minhng.info)](https://minhng.info/tutorials/xu-ly-anh-opencv-can-bang-sang-histogram-equalization.html)

**Tiếng Anh**

[4]. [statistics — Mathematical statistics functions — Python 3.11.0 documentation](https://docs.python.org/3/library/statistics.html)

[5]. [Python statistics Module (w3schools.com)](https://www.w3schools.com/python/module_statistics.asp)

[6]. [Histogram Equalization in Python from Scratch | by Tory Walker | HackerNoon.com | Medium](https://medium.com/hackernoon/histogram-equalization-in-python-from-scratch-ebb9c8aa3f23)