

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN MÔN :
XÁC SUẤT VÀ THỐNG KÊ CHO CÔNG NGHỆ THÔNG TIN**

...Bài luận giữ kỳ...

Người hướng dẫn: **GV Lê Anh Tuấn**

Người thực hiện: **Trương Thái Đan Huy**

MSSV: 52100222

Lớp : 21050301

Khoá : 25

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**



**BÀI TẬP LỚN MÔN XÁC SUẤT VÀ THỐNG KÊ CHO
CÔNG NGHỆ THÔNG TIN**

...Bài luận giữa kỳ...

Người hướng dẫn: **GV Lê Anh Tuấn**
Người thực hiện: **Trương Thái Đan Huy**
MSSV : 52100222
Lớp : 21050301
Khoá : 25

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

LỜI CẢM ƠN

Để hoàn thành bài luận giữa kỳ này, em xin gửi lời cảm ơn chân thành đến :

Ban giám hiệu trường Đại Học Tôn Đức Thắng vì đã tạo điều kiện về cơ sở vật chất với hệ thống thư viện hiện đại và thiết bị học tập tiên tiến, với các loại sách đa dạng, tài liệu thuận lợi cho em hoàn thành bài luận giữa kỳ này.

Xin cảm ơn giảng viên bộ môn Xác suất và thống kê cho công nghệ thông tin : thầy Lê Anh Tuấn giảng viên thực hành và cô Nguyễn Thị Huỳnh Trâm giảng viên lý thuyết bộ môn này đã giảng dạy tận tâm, chi tiết để em có đủ kiến thức và vận dụng vào bài luận lần này.

Do còn hạn chế về kiến thức nên bài luận sẽ không tránh khỏi những thiếu sót. Rất mong nhận được sự nhận xét, ý kiến, đóng góp, phê bình từ phía Thầy, Cô để bài luận được hoàn thiện hơn.

Lời cuối, em xin kính chúc thầy Lê Anh Tuấn và cô Nguyễn Thị Huỳnh Trâm và khoa CNTT nhiều sức khỏe, luôn hạnh phúc và đầy nhiệt huyết.

Em xin cảm ơn !

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi và được sự hướng dẫn của GV Lê Anh Tuấn;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án 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 tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 3 tháng 11 năm 2022

Tác giả

(ký tên và ghi rõ họ tên)

Trương Thái Đan Huy

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

Tp. Hồ Chí Minh, ngày tháng năm 2022
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

Tp. Hồ Chí Minh, ngày tháng năm 2022
(kí và ghi họ tên)

TÓM TẮT

Nhận được chủ đề của báo cáo lần này, em cảm thấy rất hứng thú với môn học, khi mình có thể tự tìm hiểu ra những công thức, những định lý, những hàm toán và cả một thuật toán để có thể dễ dàng trong việc thống kê, tính xác suất của các sự việc cũng như cả xử lý ảnh. Qua bài báo cáo này, đã nắm 19 hàm tính thống kê và một thuật toán xử lý ảnh. Tuy lượng kiến thức khá nhiều nhưng em đã cố gắng để tìm kiếm, tìm hiểu các thư viện các thuật toán cho bài báo cáo này.

MỤC LỤC

LỜI CẢM ƠN	i
PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN	iii
TÓM TẮT	iv
MỤC LỤC	1
CHƯƠNG 1 – Các hàm thống kê toán học.....	3
1.1 Giá trị trung bình và số đo của vị trí trung tâm.....	3
1.1.1 Hàm mean(data)	3
1.1.2 Hàm fmean(data,weight = None)	3
1.1.3 Hàm geometric_mean(data)	4
1.1.4 Hàm harmonic_mean(data, weights=None).....	5
1.1.5 Hàm median(data).....	6
1.1.6 Hàm median_low(data).....	7
1.1.7 Hàm median_high(data)	7
1.1.8 Hàm median_grouped(data, interval = 1)	8
1.1.9 Hàm mode(data)	8
1.1.10 Hàm multimode(data)	9
1.1.11 Hàm quantiles(data,*,n = 4, method = ‘exclusive’	9
1.2 Phép đo mức độ lan truyền	10
1.2.1 Hàm pstdev(data, mu = None).....	10
1.2.2 Hàm pvariance(data, mu = None).....	11
1.2.3 Hàm stdev(data, xbar = None)	12
1.2.4 Hàm Variance(data, xbar=None).....	12
1.3 Thống kê mối quan hệ giữa hai dữ liệu đầu vào	13
1.3.1 Hàm correlation(x, y, /)	13

1.3.2	Hàm covariance(x,y,/)	14
1.3.3	Hàm linear_regression(x,y,/,*,proportional = False)	14
CHƯƠNG 2	– Cân bằng histogram (Histogram equalization algorithm)	16
2.1	Tổng quát	16
2.2	Giải thuật của cân bằng histogram	16
2.3	Đánh giá	18
CHƯƠNG 3	– Implementation	18
3.1	Code	18
3.2	giải thích code	19
3.2	giải thích code	20
TÀI LIỆU THAM KHẢO	21
Tiếng việt	21
Tiếng anh	21

CHƯƠNG 1 – Các hàm thống kê toán học

Trong python, một số phép toán học có thể được thực hiện một cách dễ dàng bằng cách import các mô-đun để xác định các hàm khác nhau giúp nhiệm vụ của chúng ta dễ dàng hơn đặc biệt là trong thống kê và xử lý dữ liệu.

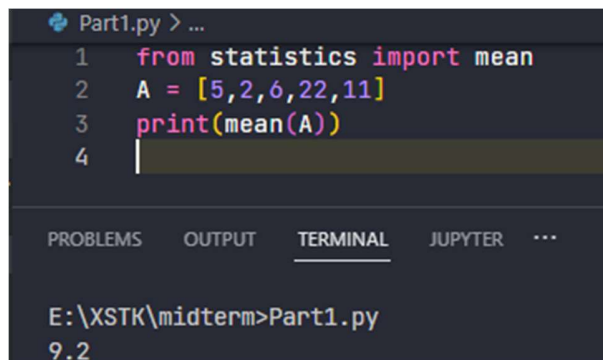
Các hàm thống kê có các chức năng hỗ trợ các kiểu dữ liệu int, float. Decimal và Fraction

1.1 Giá trị trung bình và số đo của vị trí trung tâm

1.1.1 Hàm mean(data)

- Dùng để tính trung bình số học của một dãy số học
- Ví dụ ta truyền vào mean() một mảng danh sách các số, hàm này sẽ trả về cho ta giá trị trung bình số học của dãy dữ liệu đưa vào .
- Tham số đầu vào : dữ liệu có thể là một chuỗi hoặc một bộ dữ liệu.
- Ý nghĩa kết quả đầu ra : trả về giá trị trung bình của bộ dữ liệu.
- Ví dụ : khi ta truyền vào một dãy số : 5 , 2, 6, 22, 11 thì khi sử dụng hàm mean() sẽ trả về cho ta giá trị là 9.2 có nghĩa là $(5 + 2 + 6 + 22 + 11)/5 = 9.2$
- Minh họa bằng code :

```
from statistics import mean
A = [5,2,6,22,11]
print(mean(A))
```



The screenshot shows a Jupyter Notebook interface. The top part is the code editor with the following code:

```
1 from statistics import mean
2 A = [5,2,6,22,11]
3 print(mean(A))
4
```

Below the code editor is the terminal output, which shows the command prompt and the result of the execution:

```
E:\XSTK\midterm>Part1.py
9.2
```

Bước 1 : thêm mô-đun statistics để gọi hàm mean()

Bước 2 : Thêm một danh sách giá trị vào A[]

Bước 3 : gọi hàm mean(A) để tính giá trị trung b

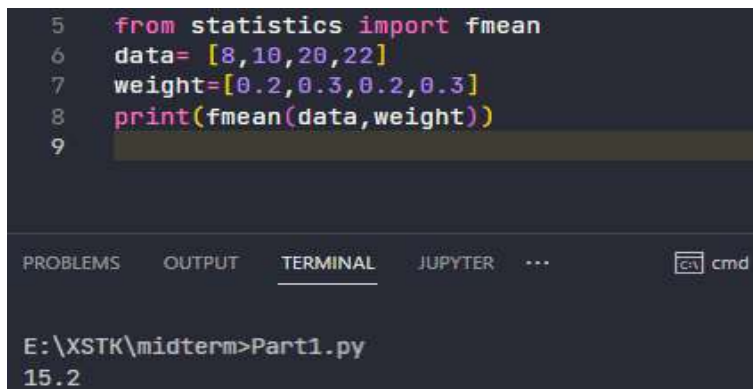
1.1.2 Hàm fmean(data,weight = None)

- Hàm fmean giống với mean() nhưng chạy nhanh hơn và luôn trả về giá trị kiểu float.


- Dữ liệu đầu vào có thể là một chuỗi hoặc lặp lại.
- Ý nghĩa tham số đầu ra : trả về giá trị trung bình của bộ dữ liệu với kiểu dữ liệu float.
- Data là dữ liệu đưa vào, weight là trọng số, tuy nhiên khi đặt trọng số, thì số lượng phần tử weight phải bằng với số lượng phần tử của dữ liệu đưa vào. Trước đây phần trọng số không được hỗ trợ, nhưng từ phiên bản 3.11 thì python đã thêm trọng số vào fmean().
- Ví dụ khi ta cần tính tổng của một dãy số, tuy nhiên dãy số chỉ lấy một lượng phần trăm nhất định như điểm môn học, lương, chiết khấu ... thì ta cần sử dụng hàm fmean().

VD : cho một dãy dữ liệu gồm 8,10,20,22 nhưng các phần tử lại lấy với tỉ lệ phần trăm như sau : 0.2, 0.3,0.2,0.3 thì kết quả sẽ như sau :

```
from statistics import fmean
data= [8,10,20,22]
weight=[0.2,0.3,0.2,0.3]
print(fmean(data,weight))
```



```
5 from statistics import fmean
6 data= [8,10,20,22]
7 weight=[0.2,0.3,0.2,0.3]
8 print(fmean(data,weight))
9
```

PROBLEMS OUTPUT TERMINAL JUPYTER ...  cmd

E:\XSTK\midterm>Part1.py
15.2

Bước 1 : thêm mô-đun statistics để gọi hàm fmean()

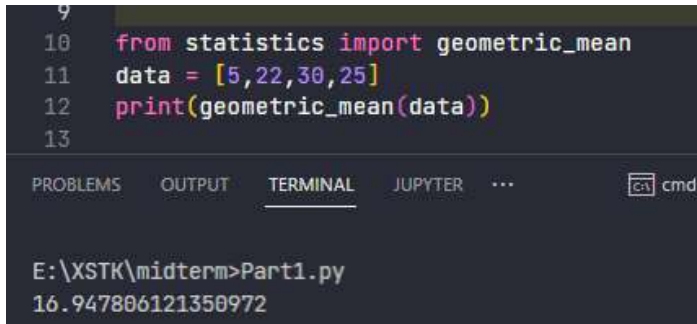
Bước 2 : tạo một danh sách giá trị data[] và danh sách trọng số weight[]

Bước 3 : gọi hàm fmean(data, weight) để tính giá trị

1.1.3 Hàm geometric_mean(data)

- Hàm tính trung bình hình học, hàm geometric sẽ chuyển dữ liệu thành dạng float và tính giá trị trung bình hình học. Có nghĩa là căn bậc số lượng phần tử của tích các phần tử đó
- Tham số đầu vào : dữ liệu có thể là một chuỗi hoặc một bộ dữ liệu.
- Hàm geometric dùng để tính xu hướng của giá trị trung tâm của dãy số.
- Ví dụ cho dãy số : 5, 22, 30, 25 tìm xu hướng trung tâm của dãy số, ta có đoạn mã như sau:

```
from statistics import geometric_mean
data = [5,22,30,25]
print(geometric_mean(data))
```



```

9
10 from statistics import geometric_mean
11 data = [5,22,30,25]
12 print(geometric_mean(data))
13
PROBLEMS OUTPUT TERMINAL JUPYTER ... cmd
E:\XSTK\midterm>Part1.py
16.947806121350972

```

Bước 1 : thêm mô-đun statistics để gọi hàm `geometric_mean()`

Bước 2 : tạo một danh sách giá trị `data[]`

Bước 3 : gọi `geometric_mean(data)` để tính xu hướng của giá trị trung tâm trong dãy số

1.1.4 Hàm `harmonic_mean(data, weights=None)`


- Gần giống với hàm `mean()`, đều trả về giá trị trung bình của dữ liệu, một chuỗi hoặc có thể lặp lại của các số có giá trị thực.
- Tham số đầu vào : dữ liệu có thể là một chuỗi hoặc một bộ dữ liệu.
- Giá trị trung bình điều hòa là nghịch đảo của hàm `arithmetic mean()`. Giá trị trung bình điều hòa là thước đo trung tâm của bộ dữ liệu. Nó thường được sử dụng để lấy tỷ lệ hoặc tỷ lệ trung bình, ví dụ như vận tốc.
- Giả sử một ô tô đi với vận tốc 60km/h trong 10km, sau đó giảm xuống còn 40km/h trong suốt 20km còn lại, tính tốc độ trung bình của ô tô. Giải :

```
from statistics import harmonic_mean
print(harmonic_mean([60,40],weights=[10,20]))
```

```

14 from statistics import harmonic_mean
15 print(harmonic_mean([60,40],weights=[10,20]))

```

PROBLEMS OUTPUT **TERMINAL** JUPYTER DEBUG CONSOLE  cmd +

Microsoft Windows [Version 10.0.22000.1165]
(c) Microsoft Corporation. All rights reserved.

E:\XSTK\midterm>Part1.py
45.0

1.1.5 Hàm median(data)

- Hàm median(data) hàm trung bị, trả về giá trị ở giữa của dữ liệu số. sử dụng phương pháp “giá trị trung bình của hai giá trị giữa”. dữ liệu có thể là một chuỗi hoặc lặp lại.
- Hàm này chỉ trả về giá trị của vị trí trung tâm, và ít sự ảnh hưởng bởi sự hiện diện của các điểm ngoại lệ.
- Khi số lượng phần tử là lẻ, thì dữ liệu ở vị trí chính giữa sẽ được trả về. Tuy nhiên nếu là chẵn thì hàm sẽ trả về giá trị trung bình của 2 giá trị tại 2 vị trí ở giữa.
- Ví dụ : tính trung vị dãy số A = [1, 5, 7] và dãy B = [1, 4, 6, 7]. Giải :

```

from statistics import median
A =[1,3,7]
B = [1,4,6,7]
print(median(A))
print(median(B))

```

```

17 from statistics import median
18 A =[1,3,7]
19 B = [1,4,6,7]
20 print(median(A))
21 print(median(B))

```

PROBLEMS OUTPUT **TERMINAL** JUPYTER ...

E:\XSTK\midterm>Part1.py
3
5.0

1.1.6 Hàm median_low(data)

- Giống với hàm median(data) nhưng nếu số lượng phần tử là chẵn thì sẽ trả về giá trị nhỏ hơn
- VD : tính trung vị thấp dãy số A = [1, 5, 7] và dãy B = [1, 4, 6, 7]

```
from statistics import median_low
A = [1, 3, 7]
B = [1, 4, 6, 7]
print(median_low(A))
print(median_low(B))
```



```
23 from statistics import median_low
24 A = [1, 3, 7]
25 B = [1, 4, 6, 7]
26 print(median_low(A))
27 print(median_low(B))
28
```

PROBLEMS OUTPUT TERMINAL JUPYTER ... cmd + v

E:\XSTK\midterm>Part1.py

3

4

E:\XSTK\midterm>

1.1.7 Hàm median_high(data)

- Hàm median_high(data) nhưng nếu số lượng phần tử là chẵn thì sẽ trả về giá trị lớn hơn.
- VD : tính trung vị cao dãy số A = [1, 5, 7] và dãy B = [1, 4, 6, 7]

```
from statistics import median_high
A = [1, 3, 7]
B = [1, 4, 6, 7]
print(median_high(A))
print(median_high(B))
```

```

29 from statistics import median_high
30 A = [1,3,7]
31 B = [1,4,6,7]
32 print(median_high(A))
33 print(median_high(B))

```

PROBLEMS OUTPUT TERMINAL JUPYTER ...

E:\XSTK\midterm>Part1.py
3
6

E:\XSTK\midterm>

1.1.8 Hàm median_grouped(data, interval = 1)

- Trả về giá trị trung bình của dữ liệu liên tục được nhóm lại, được tính bằng phân vị thứ 50 sử dụng phép nội suy.
- Tham số đầu vào : dữ liệu có thể là một chuỗi hoặc một bộ dữ liệu.

```

from statistics import median_grouped
A = [1,4,5,5,7]
print(median_grouped(A,interval=2))

```

```

38 from statistics import median_grouped
39 A = [1,4,5,5,7]
40 print(median_grouped(A,interval=2))
41

```

PROBLEMS OUTPUT TERMINAL JUPYTER ...

E:\XSTK\midterm>Part1.py
4.5

1.1.9 Hàm mode(data)

- Hàm mode(data) trả về giá trị xuất hiện nhiều lần nhất trong bộ dữ liệu.
- Tham số đầu vào : dữ liệu có thể là một chuỗi hoặc một bộ dữ liệu.
- Ví dụ cho dãy dữ liệu A = [1,2,3,2,2,7,8,4,3] tìm mode của bộ dữ liệu

```

from statistics import mode
A = [1,2,3,2,2,7,8,4,3]
print(mode(A))

```

```

42 from statistics import mode
43 A = [1,2,3,2,2,7,8,4,3]
44 print(mode(A))
45

```

PROBLEMS OUTPUT TERMINAL JUPYTER ...

```

E:\XSTK\midterm>Part1.py
2

```

1.1.10 Hàm multimode(data)

- Hàm multimode(data) dùng để trả về danh sách các giá trị thường xuyên nhất theo thứ tự lần đầu tiên chúng gặp trong bộ dữ liệu.
- Tham số đầu vào : là một list, array, tuple hoặc iterator của các số có giá trị thực.
- VD : cho chuỗi S = 'aabbbbccddddeeffffgg' tìm multimode của S:

```

from statistics import multimode
S = 'aabbbbccddddeeffffgg'
print(multimode(S))

```

```

46 from statistics import multimode
47 S = 'aabbbbccddddeeffffgg'
48 print(multimode(S))
49

```

PROBLEMS OUTPUT TERMINAL JUPYTER ...

```

E:\XSTK\midterm>Part1.py
['b', 'd', 'f']

E:\XSTK\midterm>

```

1.1.11 Hàm quantiles(data,*,n = 4, method = 'exclusive')

- Hàm số lượng tử dùng để chia dữ liệu thành n khoảng liên tục với xác suất bằng nhau. Trả về danh sách các điểm cắt tách các khoảng n – 1.
- Tham số đầu vào : là một list, array, tuple hoặc iterator của các số có giá trị thực.

-
- Đặt n thành 4 cho phân tử (mặc định). Đặt n = 10 cho các deciles. Đặt n = 100 cho các phân vị, cung cấp 99 điểm cắt để phân tách thành 100 nhóm có kích thước bằng nhau.
- Để kết quả có ý nghĩa, số lượng điểm dữ liệu phải lớn hơn n.
- Ví dụ về hàm quantiles:
Cho bộ dữ liệu data = [105, 129, 87, 86, 111, 111, 89, 81, 108, 92, 110, 100, 75, 105, 103, 109, 76, 119, 99, 91, 103, 129, 106, 101, 84, 111, 74, 87, 86, 103, 103, 106, 86, 111, 75, 87, 102, 121, 111, 88, 89, 101, 106, 95, 103, 107, 101, 81, 109, 104].

```
from statistics import quantiles
data = [105, 129, 87, 86, 111, 111, 89, 81, 108, 92, 110, 100, 75, 105, 103, 109, 76, 119, 99, 91, 103, 129, 106, 101, 84, 111, 74, 87, 86, 103, 103, 106, 86, 111, 75, 87, 102, 121, 111, 88, 89, 101, 106, 95, 103, 107, 101, 81, 109, 104]
print([round(q, 1) for q in quantiles(data, n=10)])
```

```
66 from statistics import quantiles
67 data = [105, 129, 87, 86, 111, 111, 89, 81, 108, 92, 110, 100, 75, 105, 103, 109, 76, 119, 99, 91, 103, 129, 106, 101, 84, 111, 74, 87, 86, 103, 103, 106, 86, 111, 75, 87, 102, 121, 111, 88, 89, 101, 106, 95, 103, 107, 101, 81, 109, 104]
69
70
71
72 print([round(q, 1) for q in quantiles(data, n=10)])
```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

E:\XSTK\midterm>Part1.py
[81.0, 86.2, 89.0, 99.4, 102.5, 103.6, 106.0, 109.8, 111.0]

1.2 Phép đo mức độ lan truyền

1.2.1 Hàm pstdev(data, mu = None)

- Hàm pstdev(data, mu = None) trả về độ lệch chuẩn tổng thể (có nghĩa là căn bậc hai phương sai tổng thể).
- Tham số đầu vào : là một list, array, tuple hoặc iterator của các số có giá trị thực.
- VD cho dãy A = [1.5, 2.5, 2.5, 2.75, 3.25, 4.75] tìm độ lệch chuẩn của A:


```
from statistics import pstdev
A = [1.5, 2.5, 2.5, 2.75, 3.25, 4.75]
print(pstdev(A))
```

```
50 from statistics import pstdev
51 A = [1.5, 2.5, 2.5, 2.75, 3.25, 4.75]
52 print(pstdev(A))
```

PROBLEMS OUTPUT TERMINAL JUPYTER ... cmd + v

E:\XSTK\midterm>Part1.py
0.986893273527251

E:\XSTK\midterm>

1.2.2 Hàm pvariance(data, mu = None)

- Hàm pvariance(data, mu = None) trả về phương sai tổng thể của dữ liệu, một chuỗi rỗng hoặc lặp lại các số có giá trị thực. Phương sai là thước đo tính biến thiên của dữ liệu. Phương sai nhỏ thì dữ liệu được trải rộng, còn phương sai nhỏ thì tập hơn chặt chẽ hơn xung quanh giá trị trung bình.
- Tham số đầu vào : là một list, array, tuple hoặc iterator của các số có giá trị thực.
- VD : tìm phương sai của bộ dữ liệu data = [0.0, 0.25, 0.25, 1.25, 1.5, 1.75, 2.75, 3.25]:

```
from statistics import pvariance
data = [0.0, 0.25, 0.25, 1.25, 1.5, 1.75, 2.75, 3.25]
print(pvariance(data))
```

```
53 from statistics import pvariance
54 data = [0.0, 0.25, 0.25, 1.25, 1.5, 1.75, 2.75, 3.25]
55 print(pvariance(data))
```

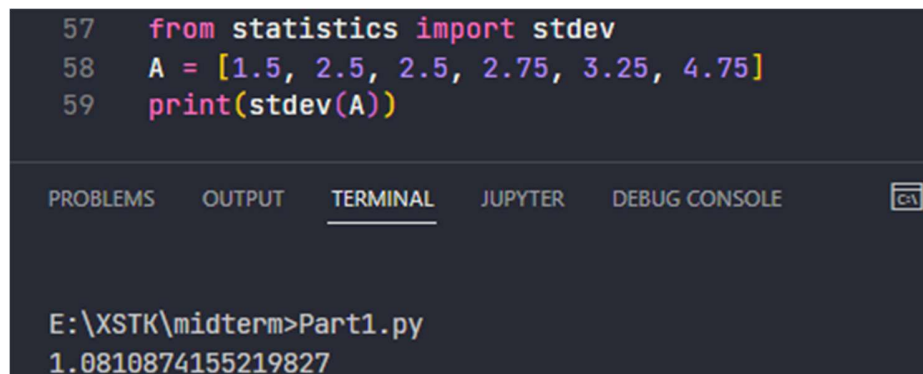
PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE cmd + v

E:\XSTK\midterm>Part1.py
1.25

1.2.3 Hàm stdev(data, xbar = None)

- Hàm stdev(data, xbar = None) trả về độ lệch chuẩn của mẫu, bằng căn bậc hai của phương sai mẫu.
- Tham số đầu vào : là một list, array, tuple hoặc iterator của các số có giá trị thực.
- VD : tìm phương sai mẫu của dãy $A = [1.5, 2.5, 2.5, 2.75, 3.25, 4.75]$

```
from statistics import stdev
A = [1.5, 2.5, 2.5, 2.75, 3.25, 4.75]
print(stdev(A))
```



```
57 from statistics import stdev
58 A = [1.5, 2.5, 2.5, 2.75, 3.25, 4.75]
59 print(stdev(A))
```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

```
E:\XSTK\midterm>Part1.py
1.0810874155219827
```

1.2.4 Hàm Variance(data, xbar=None)


- Hàm variance(data, xbar = None) trả về phương sai của mẫu, có thể lặp lại của ít nhất hai số thực. Phương sai, hay thời điểm thứ hai về giá trị trung bình, là thước đo tính biến thiên trải rộng của bộ dữ liệu. Một phương sai lớn chỉ ra rằng dữ liệu được trải rộng; một phương sai nhỏ cho thấy nó được tập hợp chặt chẽ xung quanh giá trị trung bình.
- Tham số đầu vào : là một list, array, tuple hoặc iterator của các số có giá trị thực.
- VD : tìm phương sai mẫu của bộ dữ liệu sau $A = [2.75, 1.75, 1.25, 0.25, 0.5, 1.25, 3.5]$

```
from statistics import variance
data = [2.75, 1.75, 1.25, 0.25, 0.5, 1.25, 3.5]
print(variance(data))
```

```

62  from statistics import variance
63  data = [2.75, 1.75, 1.25, 0.25, 0.5, 1.25, 3.5]
64  print(variance(data))
65

```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE  cmd +

```

E:\XSTK\midterm>Part1.py
1.3720238095238095

E:\XSTK\midterm>

```

1.3 Thống kê mối quan hệ giữa hai dữ liệu đầu vào

1.3.1 Hàm correlation(x, y, /)

- Hàm correlation dùng để trả về hệ số tương quan của Pearson cho hai đầu vào. Hệ số tương quan của Pearson nhận các giá trị từ -1 đến 1. Nó đo độ mạnh và hướng của mối quan hệ tuyến tính, trong đó +1 có nghĩa là mối quan hệ tuyến tính rất mạnh, tích cực, -1 rất mạnh, mối quan hệ tuyến tính tiêu cực và 0 không có mối quan hệ tuyến tính
- Tham số đầu vào : là một list, array, tuple hoặc iterator của các số có giá trị thực.

```

from statistics import correlation
x = [1, 2, 3, 4, 5, 6, 7, 8, 9]
y = [9, 8, 7, 6, 5, 4, 3, 2, 1]
print(correlation(x, y))

```

```

79  from statistics import correlation
80  x = [1, 2, 3, 4, 5, 6, 7, 8, 9]
81  y = [9, 8, 7, 6, 5, 4, 3, 2, 1]
82  print(correlation(x, y))

```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

```

E:\XSTK\midterm>Part1.py

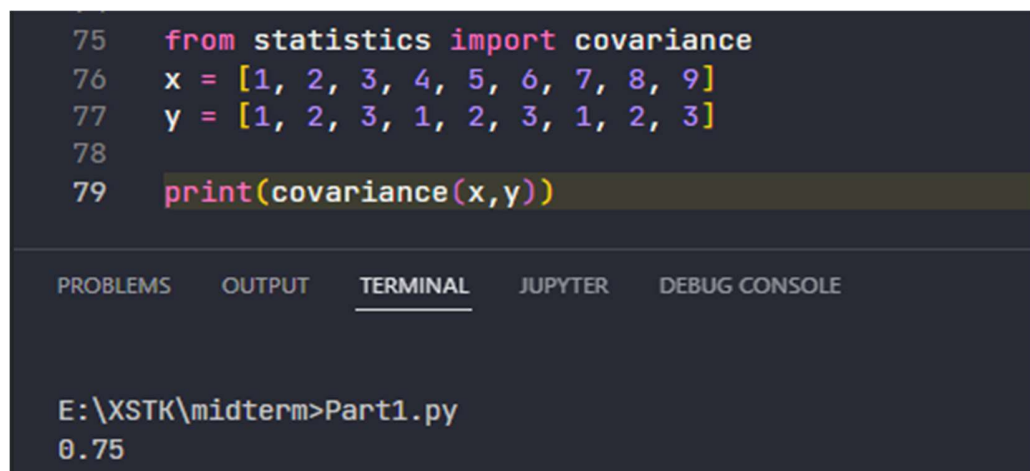
```

1.3.2 Hàm `covariance(x,y,/)`

- Hàm trả về phương sai mẫu của hai đầu vào x và y. Hiệp phương sai là thước đo sự thay đổi chung của hai đầu vào.
- Tham số đầu vào : là một list, array, tuple hoặc iterator của các số có giá trị thực.
- Cả hai đầu vào phải cùng độ dài.
- VD: cho `x = [1, 2, 3, 4, 5, 6, 7, 8, 9]`
`y = [1, 2, 3, 1, 2, 3, 1, 2, 3]`

tìm phương sai mẫu của x,y

```
from statistics import covariance
x = [1, 2, 3, 4, 5, 6, 7, 8, 9]
y = [1, 2, 3, 1, 2, 3, 1, 2, 3]
print(covariance(x,y))
```



```
75 from statistics import covariance
76 x = [1, 2, 3, 4, 5, 6, 7, 8, 9]
77 y = [1, 2, 3, 1, 2, 3, 1, 2, 3]
78
79 print(covariance(x,y))
```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

E:\XSTK\midterm>Part1.py
0.75

1.3.3 Hàm `linear_regression(x,y,/,*,proportional = False)`

- Hồi quy tuyến tính trả về độ dốc và hệ số chặn của các tham số hồi quy tuyến tính đơn giản được ước tính bằng cách sử dụng bình phương nhỏ nhất thông thường.
- Tham số đầu vào : là một list, array, tuple hoặc iterator của các số có giá trị thực.
- Cả 2 dữ liệu phải có cùng độ dài và biến độc lập x không được là hằng số.
- Ví dụ: chúng ta có thể sử dụng ngày phát hành của các bộ phim Monty Python để dự đoán số lượng tích lũy các bộ phim Monty Python sẽ được sản xuất vào năm 2019 với giả định rằng chúng đã giữ được tốc độ.

```
from statistics import linear_regression
```

```
year = [1971, 1975, 1979, 1982, 1983]
films_total = [1, 2, 3, 4, 5]
slope, intercept = linear_regression(year, films_total)
print(round(slope * 2019 + intercept))
```

```
84 from statistics import linear_regression
85 year = [1971, 1975, 1979, 1982, 1983]
86 films_total = [1, 2, 3, 4, 5]
87 slope, intercept = linear_regression(year, films_total)
88 print(round(slope * 2019 + intercept))
```

PROBLEMS

OUTPUT

TERMINAL

JUPYTER

DEBUG CONSOLE

 c

E:\XSTK\midterm>Part1.py

16

CHƯƠNG 2 – Cân bằng histogram (Histogram equalization algorithm)

2.1 Tổng quát

- Cân bằng biểu đồ là phương pháp xử lý hình ảnh điều chỉnh độ tương phản bằng cách sử dụng biểu đồ của hình ảnh.
- Phương pháp này sử dụng để làm tăng độ tương phản toàn cục của hình ảnh, đặc biệt là hình ảnh được biểu diễn bằng một dải giá trị cường độ hẹp. Bằng việc điều chỉnh các thông số giúp cho cường độ được phân bố đồng đều và đầy đủ hơn. Làm cho ảnh có độ tương phản gần như cao hơn đồng thời giảm tương phản các nơi có giá trị cường độ phổ biến cao.
- Cân bằng histogram làm cho các giá trị pixel không bị dồn lại một khoảng hẹp mà được “kéo dãn” ra.
- Các giải thuật xử lý ảnh thường nhạy cảm với ánh sáng, nhưng hình ảnh khác nhau về điều kiện sáng và môi trường có thể ảnh hưởng kết quả xử lý như trong các bài toán (phát hiện đối tượng, nhận dạng, đếm đối tượng, quét biển số xe..). Do đó bước cân bằng sáng ở tiền xử lý vô cùng quan trọng để giảm sai lệch kết quả

2.2 Giải thuật của cân bằng histogram

- Thống kê histogram cho ảnh: $H(i)$.
- Áp dụng hàm biến đổi: $Z(i) = \sum_{j=0}^i H(j)$
- Hàm biến đổi K tại một mức sáng i được tính như sau :

$$K(i) = \frac{Z(i) - \min(Z)}{\max(Z) - \min(Z)} * 255$$

Công thức này có tác dụng dãn các khoảng phân bố dày đặc pixel và co các khoảng phân bố thưa pixel.

Các thư viện sử dụng cho việc xử lý ảnh xám bao gồm : numpy, cv2, matplotlib.

Hàm tính histogram của một ảnh :

```
def compute_hist(img):
    hist = np.zeros((256,), np.uint8)
    h, w = img.shape[:2]
    for i in range(h):
        for j in range(w):
            hist[img[i][j]] += 1
    return hist
```

Hàm cân bằng histogram:

```
def equal_hist(hist):
    cumulator = np.zeros_like(hist, np.float64)
    for i in range(len(cumulator)):
        cumulator[i] = hist[:i].sum()
    print(cumulator)
    new_hist = (cumulator - cumulator.min()) / (cumulator.max() - cumulator.min()) * 255
```

Code mẫu :

```
hist = compute_hist(img).ravel()
new_hist = equal_hist(hist)

h, w = img.shape[:2]
for i in range(h):
    for j in range(w):
        img[i,j] = new_hist[img[i,j]]

fig = plt.figure()
ax = plt.subplot(121)
plt.imshow(img, cmap='gray')

plt.subplot(122)
plt.plot(new_hist)
plt.show()
```

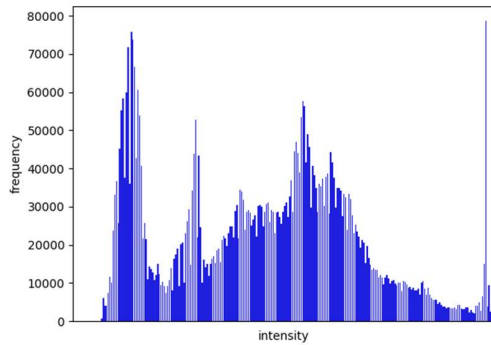
Kết quả thu được :



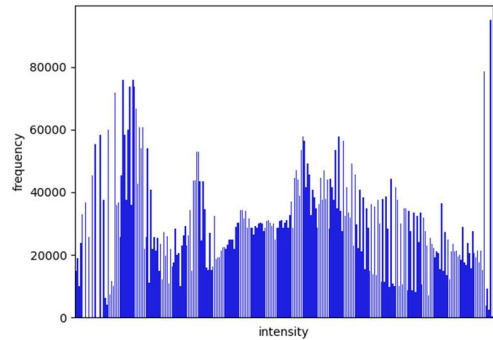
Trước khi cân bằng



Sau khi cân bằng



Trước khi cân bằng



Sau khi cân bằng

2.3 Đánh giá

Giải thuật này đã sử dụng các biểu đồ hình ảnh để phân bố lại cường độ của hình ảnh qua việc điều chỉnh các vùng tương phản cao để hạ xuống hay vùng tương phản thấp nâng cao lên. Làm cho chi tiết hình ảnh được nâng cao hơn, giúp dễ dàng kiểm soát cũng như ít trường hợp sai kết quả trong quá trình làm sử dụng dữ liệu.

CHƯƠNG 3 – Implementation

3.1 Code

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
# thêm các thư viện numpy, opencv, module pyplot từ matplotlib

img = cv2.imread("demo.jpg", 0) #import ảnh sau đó chuyển sang xám
#tạo biến img lưu hình ảnh demo.jpg
def compute_hist(img): #Hàm tính histogram của ảnh
    hist = np.zeros((256,), np.uint8) # tạo một hist cho ảnh
    h, w = img.shape[:2] # lưu chiều cao và rộng của ảnh
    for i in range(h): # duyệt chiều cao
        for j in range(w): # duyệt chiều rộng
            hist[img[i][j]] += 1 #đây là điểm ảnh của ảnh
    return hist # trả về tập điểm ảnh của ảnh

def equal_hist(hist):
```



```

    cumulator = np.zeros_like(hist, np.float64) # tạo một mảng cumulator
    có độ dài bằng độ dài mảng điểm ảnh
    for i in range(len(cumulator)): #Duyệt tất cả các điểm ảnh
        cumulator[i] = hist[:i].sum() #Lưu tổng các điểm ảnh
    print(cumulator)#in ra các điểm ảnh
    new_hist = (cumulator - cumulator.min())/(cumulator.max() -
    cumulator.min()) * 255 #Kéo giãn các điểm ảnh ra
    new_hist = np.uint8(new_hist)
    return new_hist

hist = compute_hist(img).ravel() # trả về hist một danh sách điểm ảnh
new_hist = equal_hist(hist)#cân bằng histogram

h, w = img.shape[:2] #set chiều cao và rộng cho ảnh
for i in range(h):
    for j in range(w):
        img[i,j] = new_hist[img[i,j]] #gán điểm ảnh mới đã qua xử lý vào
        điểm ảnh cũ

fig = plt.figure()
ax = plt.subplot(121)
plt.imshow(img, cmap='gray') #chỉnh ảnh xám

plt.subplot(122)
plt.plot(img)
plt.show()

```

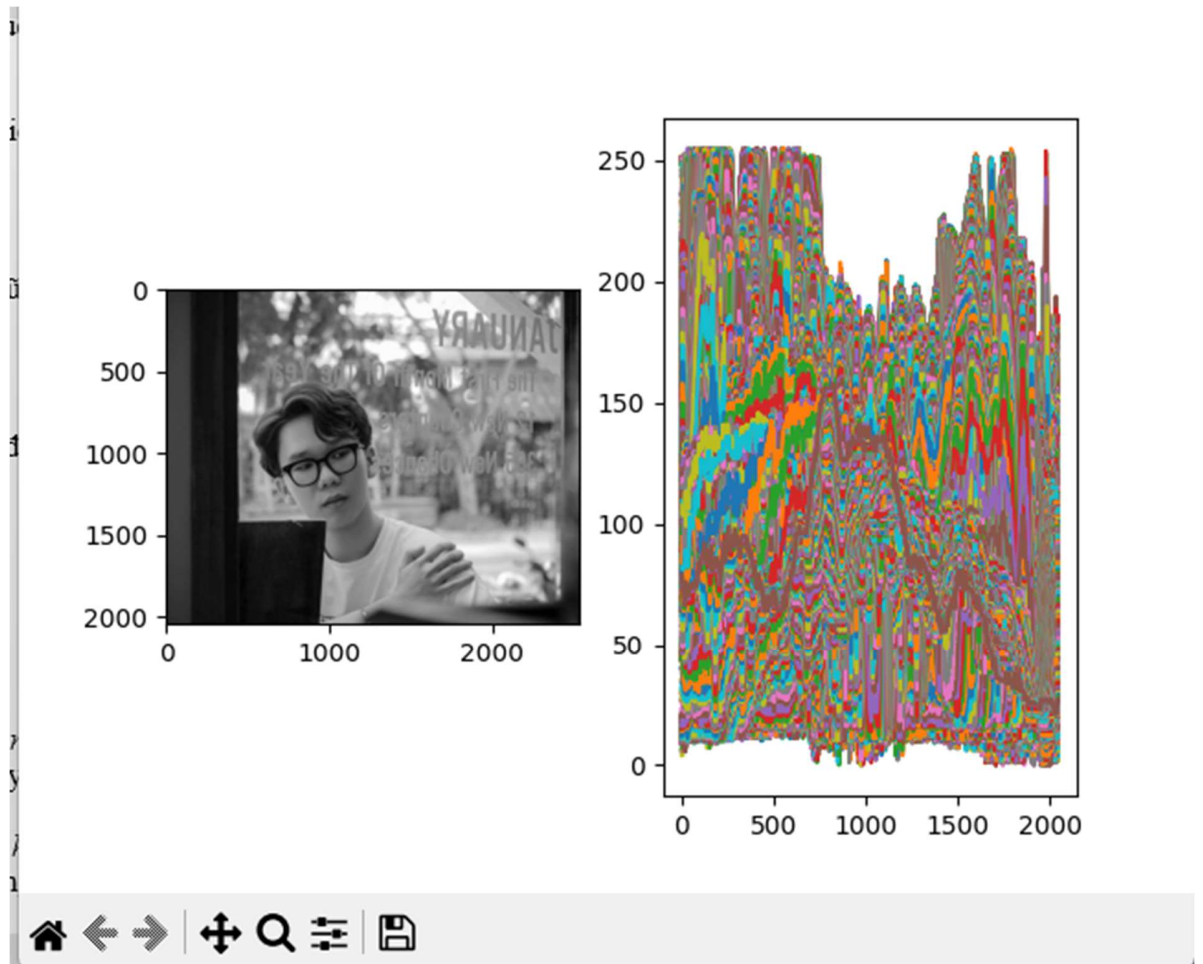
3.2 giải thích code

- Để thực hiện được giải thuật cân bằng biểu đồ ảnh, chúng ta cần 3 thư viện :
Numpy, Cv2, Matplotlib để có thể sử dụng các hàm phục vụ cho quá trình xử lý ảnh dễ dàng hơn.
- Tạo biến img để lưu trữ ảnh
- Tạo hàm compute_hist để thống kê biểu đồ của img:
 - a. Np.zeros để tạo một hist cho ảnh.
 - b. Img.shape để trả về chiều dài, rộng của ảnh.
 - c. Sau đó tính tổng các điểm ảnh lại

- Hàm `equal_hist` dùng để cân bằng các điểm ảnh lại, sao cho cục diện ảnh hài hòa nhất.
 - a. Hàm `np.zeros_like` tạo một mảng có cùng số dòng vs cột giống vs bản hist
 - b. Kéo giãn cường độ tương phản của ảnh
 - c. Trả về một hist mới đã qua chỉnh sửa
- Cuối cùng là tạo một ảnh mới, lưu những điểm ảnh đã chỉnh sửa vào trong ảnh mới. với một màu xám.

3.2 giải thích code

- Kết quả trả về một ảnh mới đã qua chỉnh sửa màu sắc, cường độ



TÀI LIỆU THAM KHẢO

Tiếng việt

Nguyen, Minh. 2018. Minhng.info. *Xử lý ảnh - OpenCV cân bằng sáng (histogram equalization)*. [Online] 10 27, 2018. <https://minhng.info/tutorials/xu-ly-anh-opencv-can-bang-sang-histogram-equalization.html>.

Nguyễn, Trung Thành. 2019. Viblo. *Xử lý ảnh: thuật toán cân bằng histogram ảnh*. [Online] 11 22, 2019. <https://viblo.asia/p/xu-li-anh-thuat-toan-can-bang-histogram-anh-GrLZDOogKk0>.

Tiếng anh

codecademy. 2022. [Online] 2022. <https://www.codecademy.com/learn/learn-statistics-with-python/modules/quartiles-quantiles-and-interquartile-range/cheatsheet>.

Foundation, The Python Software. 2022. Python. [Online] the Python Software Foundation, 11 6, 2022.

<https://docs.python.org/3/library/statistics.html#statistics.correlation>.

[blo.asia/p/xu-li-anh-thuat-toan-can-bang-histogram-anh-GrLZDOogKk0](https://viblo.asia/p/xu-li-anh-thuat-toan-can-bang-histogram-anh-GrLZDOogKk0).

Rosebrock, Adrian. 2021. pyimagesearch. *OpenCV Histogram Equalization and Adaptive Histogram Equalization (CLAHE)*. [Online] 2 1, 2021.

<https://pyimagesearch.com/2021/02/01/opencv-histogram-equalization-and-adaptive-histogram-equalization-clahe/>.