AI VIET NAM – COURSE 2023

Interpolation and its Applications - Exercises

Ngày 2 tháng 6 năm 2023

400x400

image interpolate

Formatted Date	Temperature (C)
2006-01-01 00:00:00.000 +0100	0.577778
2006-01-01 01:00:00.000 +0100	1.161111
2006-01-01 02:00:00.000 +0100	1.666667
2006-01-01 03:00:00.000 +0100	1.711111
2006-01-01 04:00:00.000 +0100	1.183333
2006-01-01 05:00:00.000 +0100	1.205556
2006-01-01 06:00:00.000 +0100	2.222222
2006-01-01 07:00:00.000 +0100	2.072222
2006-01-01 08:00:00.000 +0100	2.2
2006-01-01 09:00:00.000 +0100	**NaN**
2006-01-01 10:00:00.000 +0100	2.788889
2006-01-01 11:00:00.000 +0100	**NaN**
2006-01-01 12:00:00.000 +0100	4.911111
2006-01-01 13:00:00.000 +0100	6.205556
2006-01-01 14:00:00.000 +0100	**NaN**

interpolate()

Formatted Date	Temperature (C)
2006-01-01 00:00:00.000 +0100	0.577778
2006-01-01 01:00:00.000 +0100	1.161111
2006-01-01 02:00:00.000 +0100	1.666667
2006-01-01 03:00:00.000 +0100	1.711111
2006-01-01 04:00:00.000 +0100	1.183333
2006-01-01 05:00:00.000 +0100	1.205556
2006-01-01 06:00:00.000 +0100	2.222222
2006-01-01 07:00:00.000 +0100	2.072222
2006-01-01 08:00:00.000 +0100	2.2
2006-01-01 09:00:00.000 +0100	**2.494444**
2006-01-01 10:00:00.000 +0100	2.788889
2006-01-01 11:00:00.000 +0100	**3.85**
2006-01-01 12:00:00.000 +0100	4.911111
2006-01-01 13:00:00.000 +0100	6.205556
2006-01-01 14:00:00.000 +0100	**6.577778**

Câu hỏi 1 (Kỹ thuật thao tác trên dữ liệu danh sách 2 chiều): Cho trước một danh sách các danh sách số tự nhiên được lưu trữ trong danh sách (List) 2 chiều gồm có m dòng và n cột. Phát triển chương trình tính tổng cho mỗi dòng của danh sách. Bên dưới là ví dụ minh hoạ các test case đầu vào và đầu ra của chương trình.

```
Examples
Input:
data = [[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]

Output:
result = [6, 15, 24]
-----------------------------------
Input:
data = [[1, 2],
        [3, 4],
        [5, 6]]
Output:
[3, 7, 11]
```

```python
# implementation
def compute_sum(data):
    result = []

    # your code here

    return result

# test case
data = [[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]
assert compute_sum(data) == [6, 15, 24]

data = [[1, 2],
        [3, 4],
        [5, 6]]
assert compute_sum(data) == [3, 7, 11]

# multiple choice
data = [[6, 8],
        [3, 1],
        [7, 3]]
print(compute_sum(data))
```

```
Select one of the following answers:

a) [14, 4, 10]

b) [16, 12]

c) [14, 4, 12]

d) [14, 16, 12]
```

Câu hỏi 2 (Kỹ thuật thao tác trên dữ liệu danh sách 2 chiều):

Cho trước một danh sách các danh sách số tự nhiên được lưu trữ trong danh sách (List) 2 chiều gồm có n dòng và n cột (ma trận vuông). Phát triển chương trình hoán vị dữ liệu(thay đổi dòng thành cột và ngược lại) . Bên dưới là ví dụ minh hoạ các test case đầu vào và đầu ra của chương trình.

```
Examples
Input:
data = [[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]

Output:
result = [[1, 4, 7],
          [2, 5, 8],
          [3, 6, 9]]
----------------------------
Input:
data = [[1, 2],
        [4, 5]]

Output:
result = [[1, 4],
          [2, 5]]
```

```
# implementation
def transpose(data):
    # your code here

    return result

# test case
data = [[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]
assert transpose(data) == [[1, 4, 7],
                           [2, 5, 8],
                           [3, 6, 9]]

data = [[1, 2],
        [4, 5]]
assert transpose(data) == [[1, 4],
                           [2, 5]]

# multiple choice
data = [[7, 2],
        [3, 5]]
print(transpose(data))
```

```
Select one of the following answers:

a) [[7, 2],
    [3, 5]]

b) [[3, 7],
    [5, 2]]

c) [[7, 3],
    [2, 5]]
```

```
d) [[5, 3],
    [2, 7]]
```

Câu hỏi 3 (kỹ thuật nội suy liền kề cho dữ liệu 1 chiều (1D)): Cho trước một danh sách các số tự nhiên có chiều dài là n và một tham số k. Phát triển chương trình nội suy danh sách đầu vào thành danh sách mới có độ dài là n*k sử dụng kỹ thuật nội suy liền kề (nearest neighbor interpolation). Bên dưới là ví dụ minh hoạ các test case đầu vào và đầu ra của chương trình.

```
Examples
Input:
data = [1, 2, 3]
k = 2

Output:
result = [1, 1, 2, 2, 3, 3]
---------------------------
Input:
data = [1, 2, 3]
k = 3

Output:
result = [1, 1, 1, 2, 2, 2, 3, 3, 3]
```

```python
# implementation
def nni_1d(data, k):
    # your code here

    return result

# test case
data = [1, 2, 3]
k = 2
assert nni_1d(data, k) == [1, 1, 2, 2, 3, 3]

data = [1, 2, 3]
k = 3
assert nni_1d(data, k) == [1, 1, 1, 2, 2, 2, 3, 3, 3]

# multiple choice
data = [5, 7]
k = 3
print(nni_1d(data, k))
```

```
Select one of the following answers:

a) [5, 5, 7, 7]

b) [5, 5, 5, 7, 7, 7]

c) [5, 5, 5]

d) [7, 7, 7]
```

Câu hỏi 4 (kỹ thuật nội suy liền kề cho dữ liệu 2 chiều (2D)): Cho trước một danh sách các số tự nhiên (m dòng và n cột) và tham số k. Phát triển chương trình nội suy danh sách đầu vào thành danh sách mới (m dòng và n*k cột) sử dụng kỹ thuật nội suy liền kề (nearest neighbor interpolation). Bên dưới là ví dụ minh hoạ các test case đầu vào và đầu ra của chương trình.

```
Examples
Input:
data = [[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]
k = 2

Output:
result = [[1, 1, 2, 2, 3, 3],
          [4, 4, 5, 5, 6, 6],
          [7, 7, 8, 8, 9, 9]]
---------------------------------
Input:
data = [[1, 2],
        [4, 5],
        [7, 8]]
k = 2

Output:
result = [[1, 1, 2, 2],
          [4, 4, 5, 5],
          [7, 7, 8, 8]]
```

```python
# implementation
def h_nni_2d(data, k):
    # your code here
    return result

# test case
data = [[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]
k = 2
assert h_nni_2d(data, k) == [[1, 1, 2, 2, 3, 3],
                             [4, 4, 5, 5, 6, 6],
                             [7, 7, 8, 8, 9, 9]]

data = [[1, 2],
        [4, 5],
        [7, 8]]
k = 2
assert h_nni_2d(data, k) == [[1, 1, 2, 2],
                             [4, 4, 5, 5],
                             [7, 7, 8, 8]]

# multiple choice
data = [[3],
        [4]]
k = 3
print(h_nni_2d(data, k))
```

```
Select one of the following answers:

a) [[3, 3],
    [4, 4],
    [5, 5]]

b) [[3, 3, 3],
    [4, 4, 4]]

c) [[4, 4, 4],
    [3, 3, 3]]

d) [[3, 3, 3]]
```

Câu hỏi 5 (kỹ thuật nội suy liền kề cho dữ liệu 2 chiều (2D)): Cho trước một danh sách các số tự nhiên (m dòng và n cột) và tham số k. Phát triển chương trình nội suy dữ liệu cột của danh sách đầu vào thành danh sách mới (m*k dòng và n cột) sử dụng kỹ thuật nội suy liền kề (nearest neighbor interpolation). Bên dưới là ví dụ minh hoạ các test case đầu vào và đầu ra của chương trình.

```
Examples
Input:
data = [[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]
k = 2

Output:
result = [[1, 2, 3],
          [1, 2, 3],
          [4, 5, 6]
          [4, 5, 6],
          [7, 8, 9],
          [7, 8, 9]]
-------------------------
Input:
data = [[1, 2],
        [4, 5],
        [7, 8]]
k = 2

Output:
result = [[1, 2],
          [1, 2],
          [4, 5],
          [4, 5],
          [7, 8],
          [7, 8]]
```

```python
# implementation
def v_nni_2d(data, k):
    # your code here

    return result

# test case
data = [[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]
k = 2
```

```
assert v_nni_2d(data, k) ==  [[1, 2, 3],
                              [1, 2, 3],
                              [4, 5, 6],
                              [4, 5, 6],
                              [7, 8, 9],
                              [7, 8, 9]]

data = [[1, 2],
        [4, 5],
        [7, 8]]
k = 2
assert v_nni_2d(data, k) == [[1, 2],
                             [1, 2],
                             [4, 5],
                             [4, 5],
                             [7, 8],
                             [7, 8]]

# multiple choice
data = [[3, 7]]
k = 2
print(v_nni_2d(data, k))
```

```
Select one of the following answers:

a) [[7, 3, 7, 3]]

b) [[7, 7, 3, 3]]

c) [[3, 7, 3, 7]]

d) [[3, 3, 7, 7]]
```

Câu hỏi 6 (kỹ thuật nội suy liền kề cho dữ liệu 2 chiều (2D)): Cho trước một danh sách các số tự nhiên (m dòng và n cột) và tham số k. Phát triển chương trình nội suy dữ liệu danh sách đầu vào thành danh sách mới (m*k dòng và n*k cột) sử dụng kỹ thuật nội suy liền kề (nearest neighbor interpolation). Bên dưới là ví dụ minh hoạ các test case đầu vào và đầu ra của chương trình.

```
Examples
Input:
data = [[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]
k = 2

Output:
result = [[1, 1, 2, 2, 3, 3],
          [1, 1, 2, 2, 3, 3],
          [4, 4, 5, 5, 6, 6],
          [4, 4, 5, 5, 6, 6],
          [7, 7, 8, 8, 9, 9],
          [7, 7, 8, 8, 9, 9]]
------------------------------
Input:
data = [[1, 2],
        [4, 5],
        [7, 8]]
k = 2
```

```
Output:
result = [[1, 1, 2, 2],
          [1, 1, 2, 2],
          [4, 4, 5, 5],
          [4, 4, 5, 5],
          [7, 7, 8, 8],
          [7, 7, 8, 8]]
```

```python
# implementation
def nni_2d(data, k):
    # your code here

    return result

# test case
data = [[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]
k = 2
assert nni_2d(data, k) == [[1, 1, 2, 2, 3, 3],
                           [1, 1, 2, 2, 3, 3],
                           [4, 4, 5, 5, 6, 6],
                           [4, 4, 5, 5, 6, 6],
                           [7, 7, 8, 8, 9, 9],
                           [7, 7, 8, 8, 9, 9]]

data = [[1, 2],
        [4, 5],
        [7, 8]]
k = 2
assert nni_2d(data, k) == [[1, 1, 2, 2],
                           [1, 1, 2, 2],
                           [4, 4, 5, 5],
                           [4, 4, 5, 5],
                           [7, 7, 8, 8],
                           [7, 7, 8, 8]]

# multiple choice
data = [[1, 2, 3, 4]]
k = 2
print(nni_2d(data, k))
```

```
Select one of the following answers:

a) [[1, 1, 2, 2, 3, 3, 4, 4],
    [1, 1, 2, 2, 3, 3, 4, 4]]

b) [[1, 2, 3, 4],
    [1, 2, 3, 4]]

c) [[1, 1, 2, 2, 3, 3, 4, 4]]

d) [[1, 1, 2, 2],
    [1, 1, 2, 2]]
```

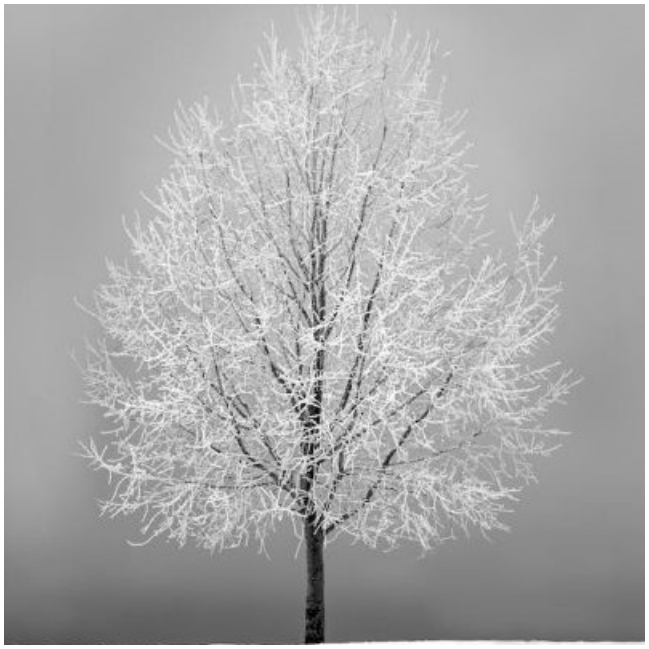
Câu hỏi 7 (Kỹ thuật nội suy linear cho dữ liệu 1 chiều (1D)): Cho trước danh sách số thực có chiều dài n và tham số k. Phát triển chương trình nội suy danh sách đầu vào thành danh sách mới có độ dài là n*k sử dụng kỹ thuật nội suy linear. Lưu ý rằng, để đơn giản, bạn có thể bỏ qua phần tử cuối cùng (k-1). Bên dưới là ví dụ minh hoạ các test case đầu vào và đầu ra của chương trình.

```
Examples
Input:
data = [1, 2, 3]
k = 2

Output:
result = [1.0, 1.5, 2.0, 2.5, 3.0, x]

---------------------------------------
Input:
data = [1, 4, 7]
k = 3

Output:
result = [1, 2.0, 3.0, 4, 5.0, 6.0, 7, x, x]

Here, x is any number.
```

```python
# implementation
import math

def linear_interpolation(p1, p2, x):
    x1, y1 = p1
    x2, y2 = p2
    if x1 != x2:
        y = ((x2-x)/(x2-x1))*y1 + ((x-x1)/(x2-x1))*y2
    else:
        y = y1
    return y

def li_1d(data, k):
    # your code here

    return result

# test case
data = [1, 2, 3]
k = 2
assert li_1d(data, k)[:-1] == [1, 1.5, 2, 2.5, 3, 0][:-1]


data = [1, 4, 7]
k = 3
assert li_1d(data, k)[:-2] == [1, 2.0, 3.0, 4, 5.0, 6.0, 7, 0, 0][:-2]

# multiple choice
data = [7, 10]
k = 3
print(li_1d(data, k))
```

```
Select one of the following answers:

a) [10.0, 9.0, 8.0, 7.0]

b) [7.0, 10.0, 13.0, 16.0]

c) [7.0, 8.0, 9.0, 10.0]

d) [7.0, 7.0, 10.0, 10.0]
```

Câu hỏi 8 (optional): Cho trước ảnh đầu vào (link1 hoặc link2) có kích thước dài 400 và rộng 400 như bên dưới:



Bạn có thể sử dụng thư viện opencv để đọc ảnh đầu và chuyển dữ liệu ảnh đầu vào thành danh sách 2 chiều (400 dòng và 400 cột).

```
import cv2
import numpy as np
import math

def resize_nni(source, source_h, source_w, target_h, target_w):
    new_data = [[0]*target_w for _ in range(target_h)]

    #Calculate horizontal and vertical scaling factor
    w_scale_factor = source_w/target_w
    h_scale_factor = source_h/target_h

    #your code here


    return new_data

image = cv2.imread('tree.jpg', 0).tolist()

height = len(image)
width = len(image[0])

new_image = resize_nni(image, height, width, height*3, width*3)
cv2.imwrite('tree_2x.jpg', np.array(new_image))
```

Ví dụ, *image* có kiểu dữ liệu là 2 chiều (a list of lists). Phát triển chương trình để tăng kích thước ảnh đầu vào (400x400) và thành ảnh mới (1200 x1200) sử dụng kỹ thuật nội suy nearest-neighbor hoặc linear.