AI VIET NAM
@aivietnam.edu.vn

Data Structure Review

List, Tuple, Set and Dictionary

Vinh Dinh Nguyen
PhD in Computer Science

Outline

- Review: Tuples vs. Lists, 2D Lists
- IoU Calculation using Tuples
- NMS Using List and Tuples
- Review: Set and Dictionary
- One-Hot Encoding Using Set and Dictionary
- Review: Tree & K-DTree
- K-D Tree Implementation

AI VIET NAM
@aivietnam.edu.vn

1D List Review

❖ A container that can contain elements

list_name = [element-1, …, element-n]

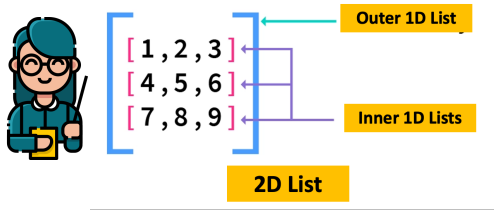
```
// create a list
data = [6, 5, 7, 1, 9, 2]
```

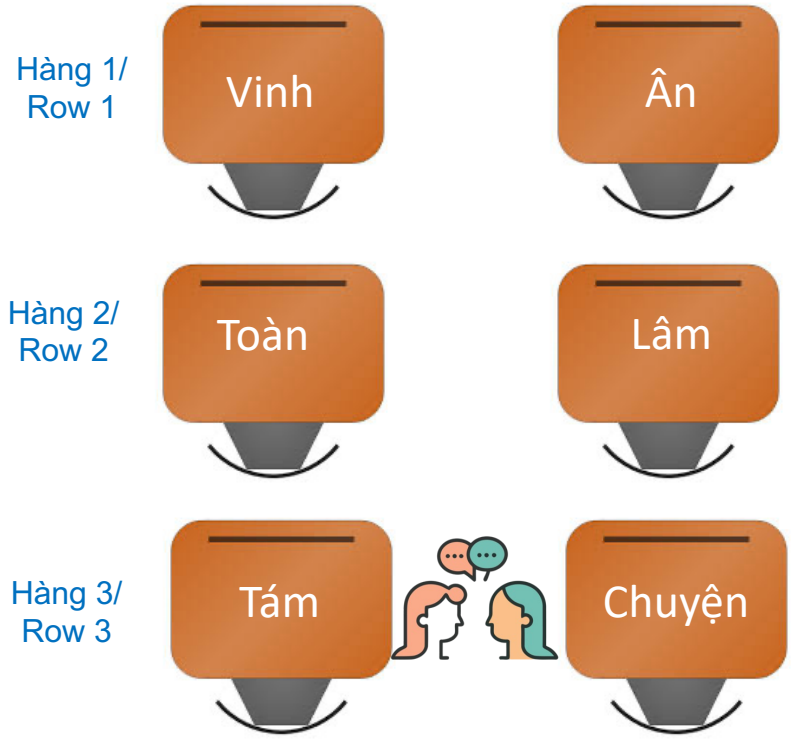
❑ Imputable

```
1.  # danh sách trống
2.  emty_list = []
3.
4.  # danh sách số tự nhiên nhỏ hơn 10
5.  my_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
6.
7.  # danh sách kết hợp nhiều kiểu dữ liệu
8.  mixedList = [True, 5, 'some string', 123.45]
9.  n_list = ["Happy", [2,0,1,5]]
10.
11. #danh sách các loại hoa quả
12. shoppingList = ['táo', 'chuối', 'cherries', 'dâu', 'mận']
```

AI VIET NAM
@aivietnam.edu.vn

From 1D to 2D List





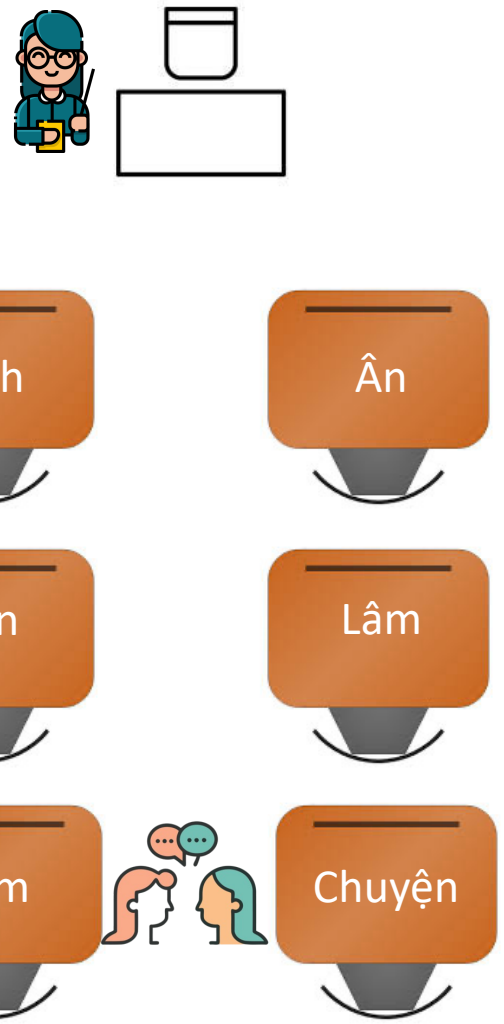
Given that the teacher cannot remember students' names, please develop a program to help her find a student's name based on his/her seat location in the class.

"Cô mời 2 bạn: ngồi ở vị trí hàng thứ 3, dãy thứ 1, và ngồi ở vị trí hàng thứ 3, dãy thứ 2" lên bảng giải thích về 2D List

"Cô mời bạn Tám và bạn Chuyện" lên bảng giải thích về 2D List

AI VIET NAM
@aivietnam.edu.vn

From 1D to 2D List




```python
student_list = ["Vinh", "An", "Toan", "Lam", "Tam", "Chuyen"]
row_class = 3
col_class = 2
def find_student(student_list, r, c):
    index = (r-1)*col_class + (c-1)
    return student_list[index]

#Find student a location (3, 1)
print(find_student(student_list, 3, 1))

#Find student a location (3, 2)
print(find_student(student_list, 3, 2))
```

```
Tam
Chuyen
```

Are there any easier solutions? This one is really difficult for newcomers

From 1D to 2D List

Progrogramer View

	Column index 0	Column index 1
Row index 0	Vinh	An
Row index 1	Toan	Lam
Row index 2	Tam	Chuyen

Teacher View

	Column 1	Column 2
Row 1	Vinh	An
Row 2	Toan	Lam
Row 3	Tam	Chuyen

Use 1D List	Vinh	An
Use 1D List	Toan	Lam
Use 1D List	Tam	Chuyen

student_list = [element1, element2, element3)

AI VIET NAM
@aivietnam.edu.vn

From 1D to 2D List

Teacher View

	Column 1	Column 2
Row 1	Vinh	An
Row 2	Toan	Lam
Row 3	Tam	Chuyen

	Column index 0	Column index 1
Row index 0	Vinh	An
Row index 1	Toan	Lam
Row index 2	Tam	Chuyen

Progrogramer View

```
row1_student = ["Vinh", "An"] #1D List
row2_student = ["Toan", "Lam"] #1D List
row3_student = ["Tam", "Chuyen"] #1D List

student_list = [row1_student, row2_student, row3_student] # 2D List
```

How to access element:
row 0 and column 1
row 0 and column 2

row1_retrieve = student_list[0] → Vinh | An

row1_retrieve[0] → Vinh

row1_retrieve[1] → An

student_list[0][0] → Vinh

student_list[0]1] → An

From 1D to 2D List

	Column 1	Column 2
Row 1	Vinh	An
Row 2	Toan	Lam
Row 3	Tam	Chuyen

Teacher View

	Column index 0	Column index 1
Row index 0	Vinh	An
Row index 1	Toan	Lam
Row index 2	Tam	Chuyen

Progrogramer View

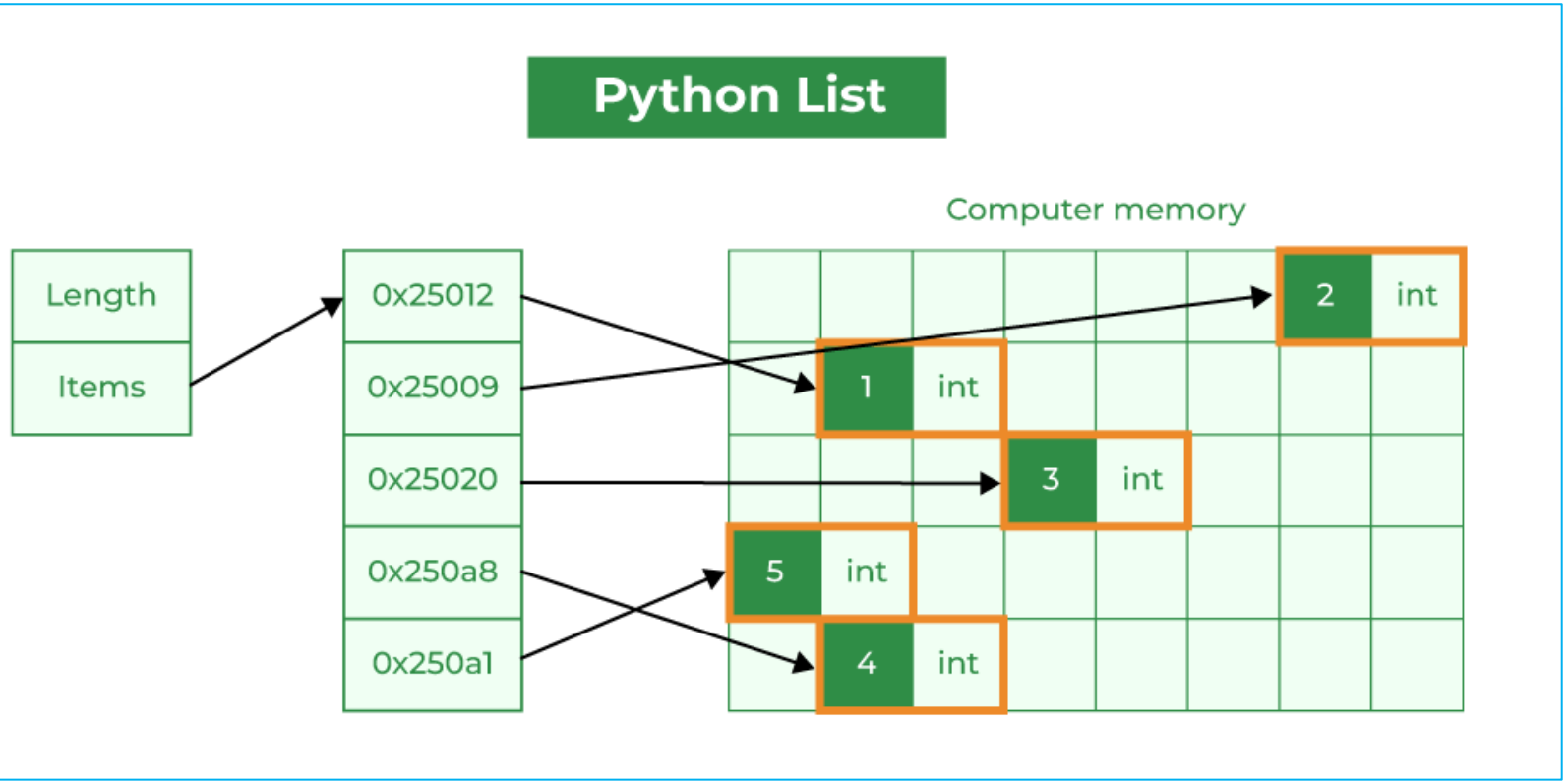
```python
row1_student = ["Vinh", "An"] #1D List
row2_student = ["Toan", "Lam"] #1D List
row3_student = ["Tam", "Chuyen"] #1D List

student_list = [row1_student, row2_student, row3_student] # 2D List

num_row = len(student_list)
num_col = len(student_list[0])

def find_student(student_list, r, c):
    return student_list[r-1][c-1]

#Find student a location (3, 1)
print(find_student(student_list, 3, 1))

#Find student a location (3, 2)
print(find_student(student_list, 3, 2))
```

```
Tam
Chuyen
```

Thanks! This one is easy to understand

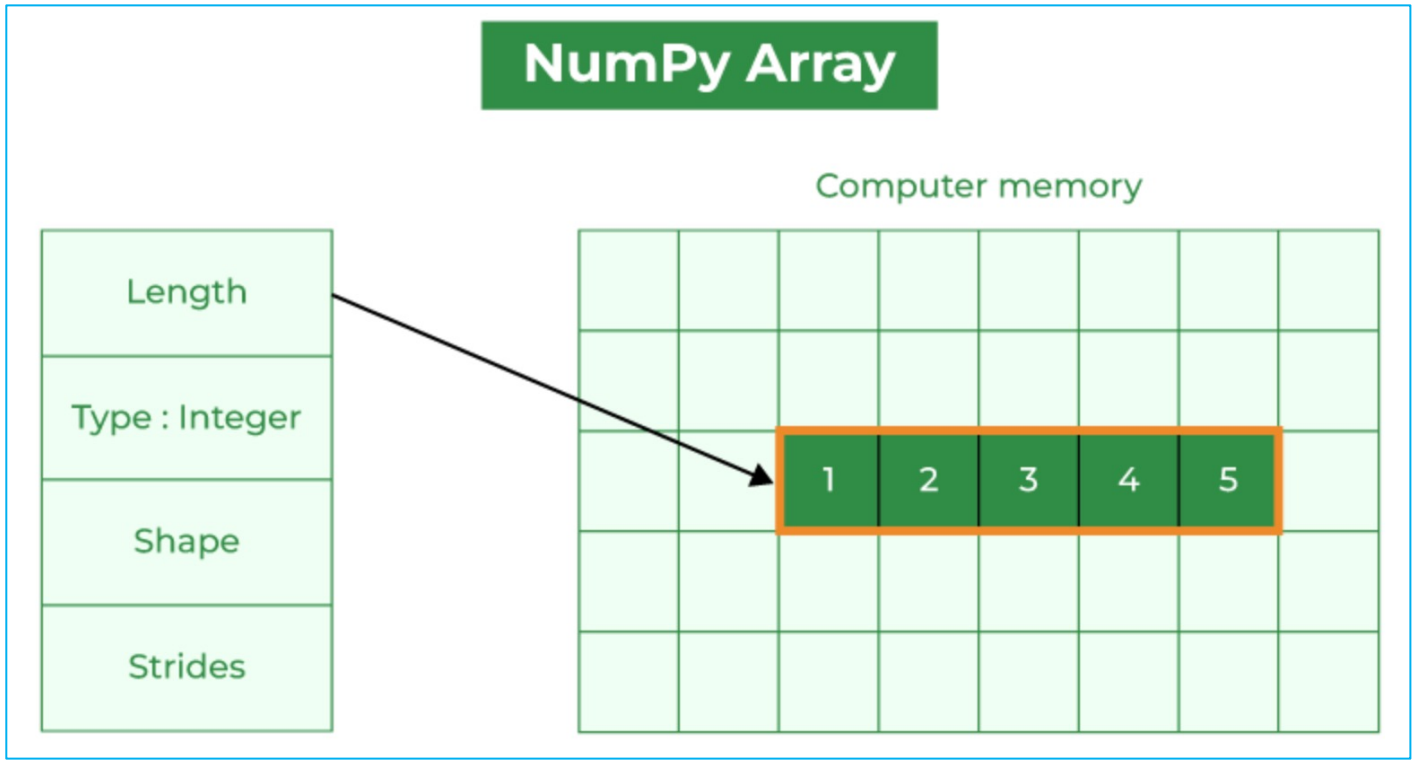
AI VIET NAM
@aivietnam.edu.vn

Python List Limitations



1.Element Overhead
2.Datatype
3.Memory Fragmentation
4.Performance
5.Functionality

NumPy Array Motivation



1.Homogeneous Data
2.Fixed Data Type
3.Contiguous Memory
4.Performance

AI VIET NAM
@aivietnam.edu.vn

1D List/Array Multiplication

array_name = **np.array**[element-1, …, element-n]

list_name = [element-1, …, element-n]

```python
def multiplizeList(size=1000):
  # declaring lists
  list1 = range(size)
  list2 = range(size)

  # capturing time before the multiplication of Python lists
  initialTime = time.time()

  # multiplying elements of both the lists and stored in another list
  resultantList = [(a * b) for a, b in zip(list1, list2)]

  # calculating execution time
  print("Time taken by Lists to perform multiplication:",
    (time.time() - initialTime),
    "seconds")
```

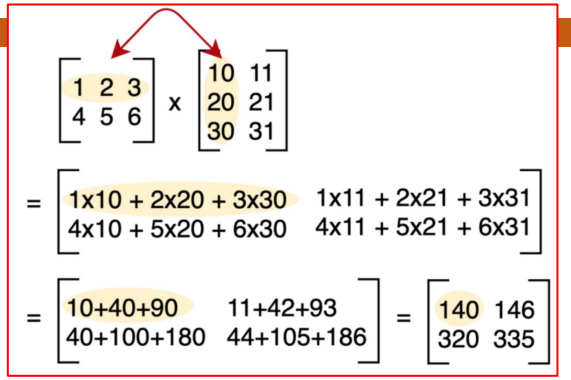
```python
def multiplizeArray(size=1000):
  # declaring arrays
  array1 = numpy.arange(size)
  array2 = numpy.arange(size)

  # capturing time before the multiplication of Numpy arrays
  initialTime = time.time()

# multiplying elements of both the Numpy arrays and stored in another Numpy array
  resultantArray = array1 * array2

# calculating execution time
  print("Time taken by NumPy Arrays to perform multiplication:",
    (time.time() - initialTime),
    "seconds")
```

```
multiplizeList(100000)
multiplizeArray(100000)
```

```
Time taken by Lists to perform multiplication: 0.025510549545288086 seconds
Time taken by NumPy Arrays to perform multiplication: 0.0005364418029785156 seconds
```

AI VIET NAM
@aivietnam.edu.vn

2D List/Array Multiplication



❖ Python List Solution

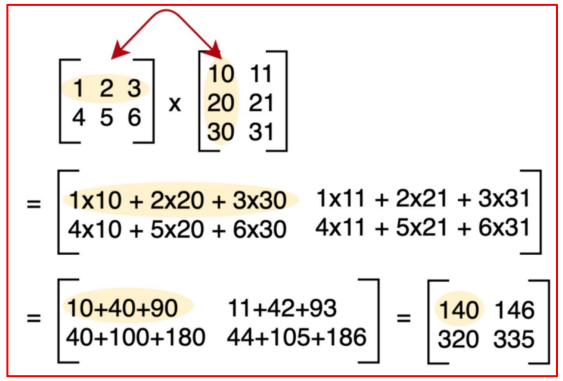
```python
def matrix_multiply(A, B):
    # Get the number of rows and columns for the input matrices
    rows_A, cols_A = len(A), len(A[0])
    rows_B, cols_B = len(B), len(B[0])

    # Ensure the number of columns in A is equal to the number of rows in B
    if cols_A != rows_B:
        raise ValueError("Number of columns in A must be equal to number of rows in B")

    # Initialize the result matrix with zeros
    result = [[0 for _ in range(cols_B)] for _ in range(rows_A)]

    # Perform the matrix multiplication
    for i in range(rows_A):
        for j in range(cols_B):
            for k in range(cols_A):
                result[i][j] += A[i][k] * B[k][j]

    return result
```

```python
# Example matrices
A = [[1, 2, 3],
     [4, 5, 6]]

B = [[7, 8],
     [9, 10],
     [11, 12]]

# Multiply the matrices
result = matrix_multiply(A, B)

# Print the result
for row in result:
    print(row)
```

```
[58, 64]
[139, 154]
```

2D List/Array Multiplication

❖ NumPy Array Solution




```python
import numpy as np

# Define the matrices
A = np.array([
    [1, 2, 3],
    [4, 5, 6]
])

B = np.array([
    [7, 8],
    [9, 10],
    [11, 12]
])

# Multiply the matrices using np.dot
result = np.dot(A, B)

# Alternatively, you can use the @ operator
# result = A @ B

# Print the result
print(result)
```

Tuple Review

- Immutability
- Performance
- Multiple Return Values
- Unpacking

❖ Structure

tuple_name = (element-1, …, element-n)

( ) can be removed

```
1.  t = 1, 2
2.  print(t)
```

```
(1, 2)
```

Tuple with one element

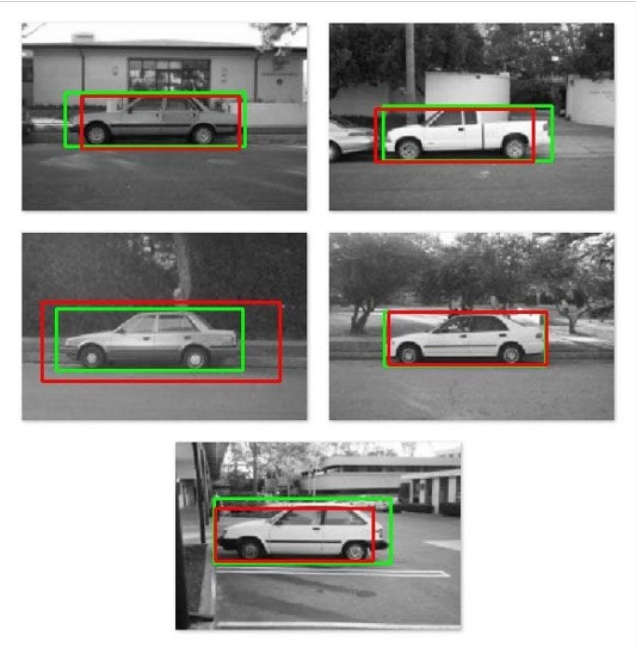
```
1  var1 = (1 + 2) * 5
2  print(type(var1), ' ', var1)
3
4  var2 = (1)
5  print(type(var2), ' ', var2)
6
7  var3 = (1,)
8  print(type(var3), ' ', var3)
```

```
<class 'int'>   15
<class 'int'>   1
<class 'tuple'>   (1,)
```

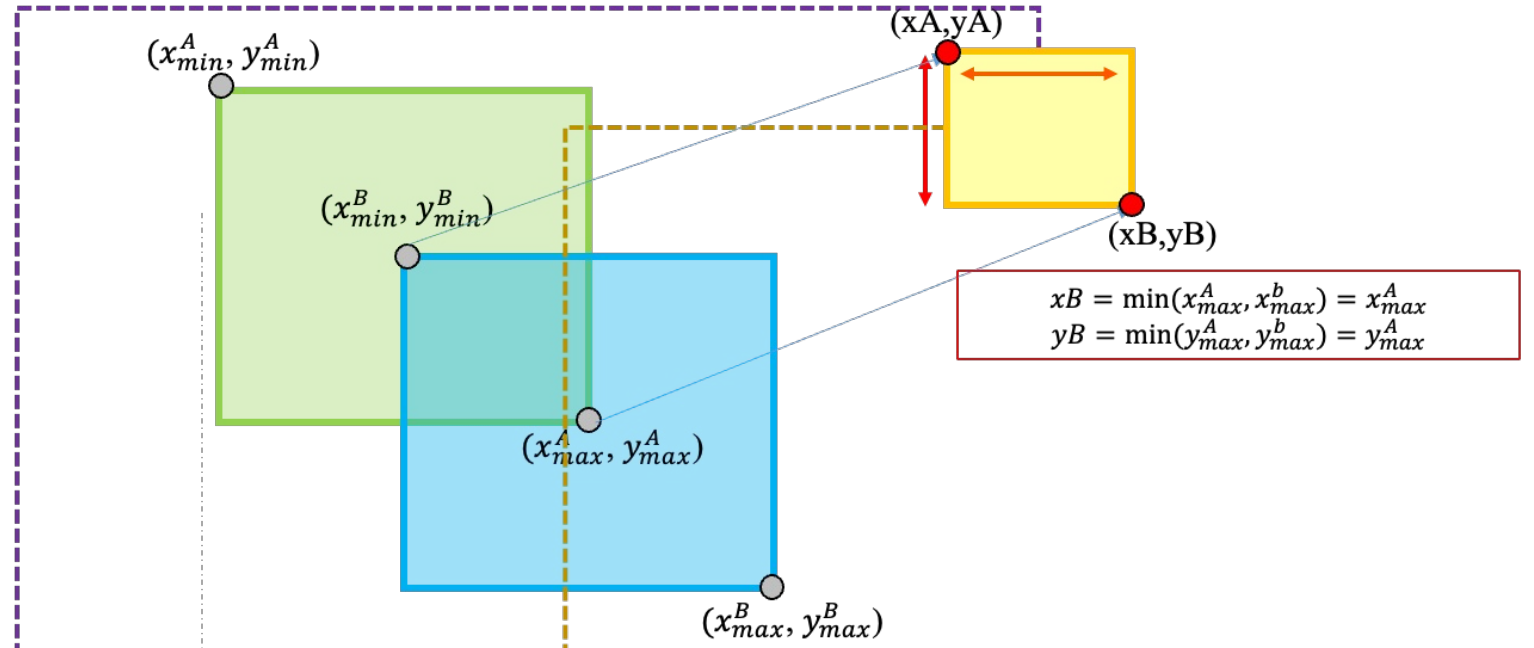
Outline

- Review: Tuples vs. Lists, 2D Lists
- IoU Calculation using Tuples
- NMS Using List and Tuples
- Review: Set and Dictionary
- One-Hot Encoding Using Set and Dictionary
- Review: Tree & K-DTree
- K-D Tree Implementation

IOU Calculation Using Tuple



$$xA = \max(x^A_{min}, x^b_{min}) = x^b_{min}$$
$$yA = \max(y^A_{min}, y^b_{min}) = y^b_{min}$$

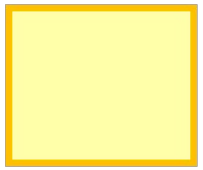


$$xB = \min(x^A_{max}, x^b_{max}) = x^A_{max}$$
$$yB = \min(y^A_{max}, y^b_{max}) = y^A_{max}$$

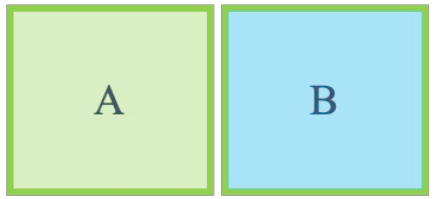
$$W = xB - xA$$

$$H = yB - yA$$

$$A \cap B = W * H$$

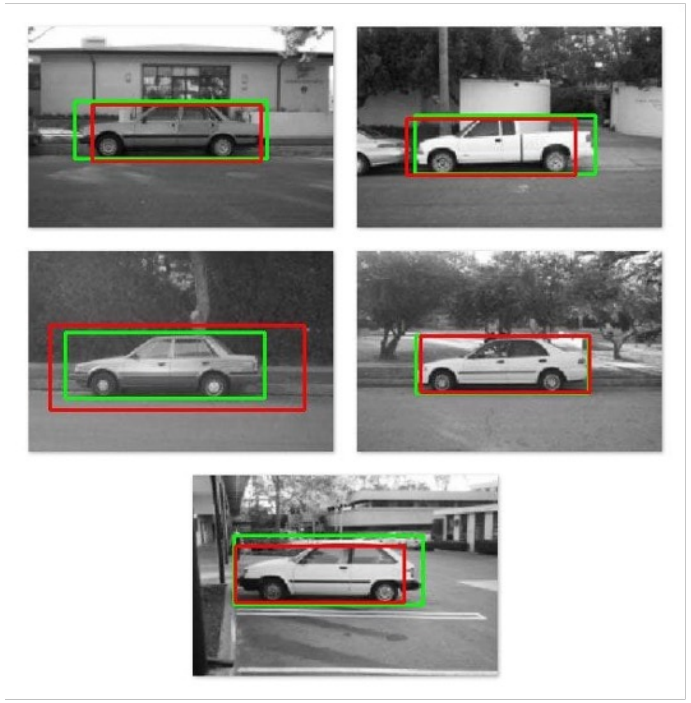


Area



AI VIET NAM
@aivietnam.edu.vn

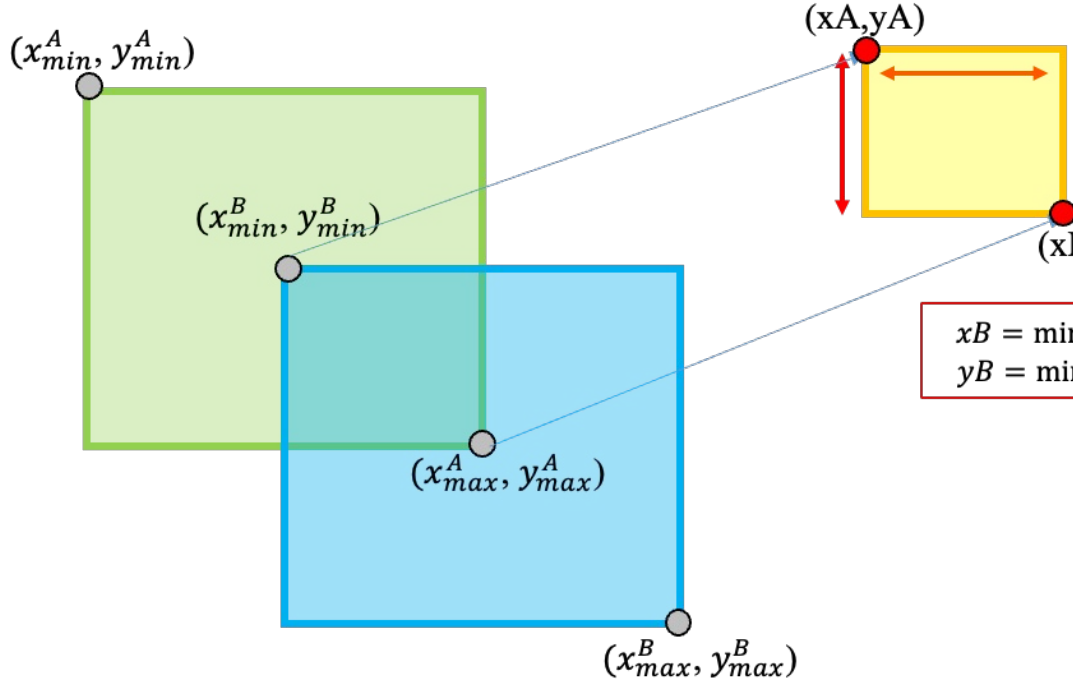
IOU Calculation Using Tuple



Giả sử ta có 2 boxes với thông tin như sau:

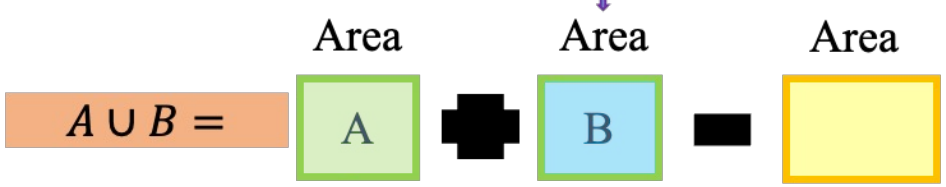
- **Box A** có tọa độ $(x^A_{min}, y^A_{min}, x^A_{max}, y^A_{max})$ => Tuple
- **Box B** có tọa độ $(x^B_{min}, y^B_{min}, x^B_{max}, y^B_{max})$ => Tuple

$$xA = \max(x^A_{min}, x^b_{min}) = x^b_{min}$$
$$yA = \max(y^A_{min}, y^b_{min}) = y^b_{min}$$



$$xB = \min(x^A_{max}, x^b_{max}) = x^A_{max}$$
$$yB = \min(y^A_{max}, y^b_{max}) = y^A_{max}$$

$$(x^B_{max} - x^B_{min}) * (y^B_{max} - y^B_{min})$$

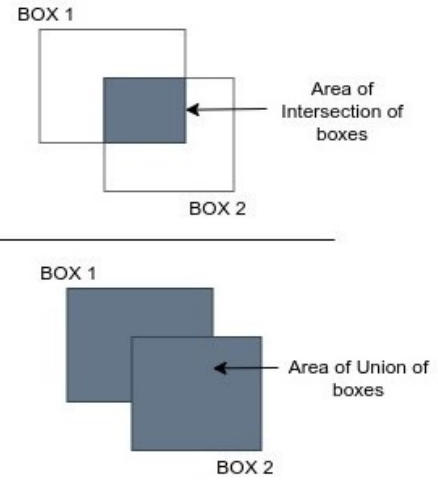


$$(x^A_{max} - x^A_{min}) * (y^A_{max} - y^A_{min})$$

AI VIET NAM
@aivietnam.edu.vn

IOU Calculation Using Tuple

IOU =



$$\frac{|A \cap B|}{|A \cup B|}$$

Tính diện tích của 2 box A, B

area1 = $(x^A_{max} - x^A_{min}) * (y^A_{max} - y^A_{min})$

area2 = $(x^B_{max} - x^B_{min}) * (y^B_{max} - y^B_{min})$

Tìm tọa đọ của vùng giao nhau (Intersection)

$$xA = \max(x^A_{min}, x^b_{min}) = x^b_{min}$$
$$yA = \max(y^A_{min}, y^b_{min}) = y^b_{min}$$

$$xB = \min(x^A_{max}, x^b_{max}) = x^A_{max}$$
$$yB = \min(y^A_{max}, y^b_{max}) = y^A_{max}$$

Tính diện tích vùng giao nhau

$W = xB - xA$

H = yB – yA

intersection_area = w*h

Tính diện tích phần hợp nhau

union_area = area1 + area2 - intersection_area

Dựa trên phần giao và phần hợp để tính IoU

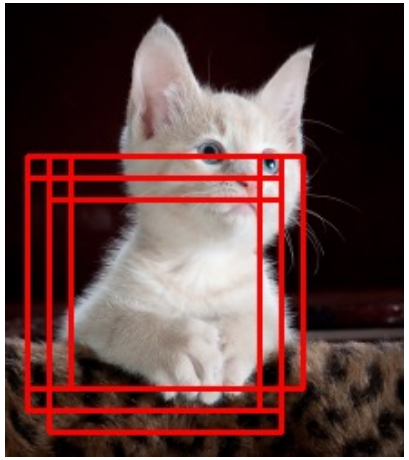
IoU = intersection_area / union_area

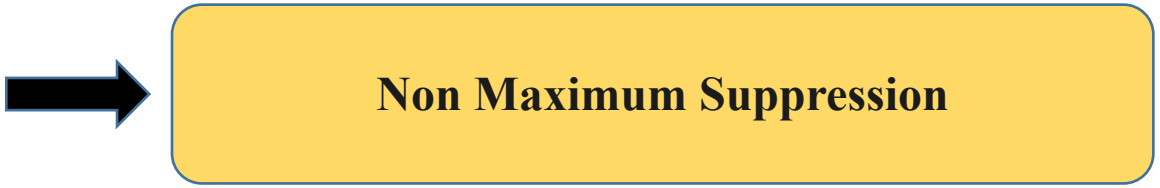
Outline

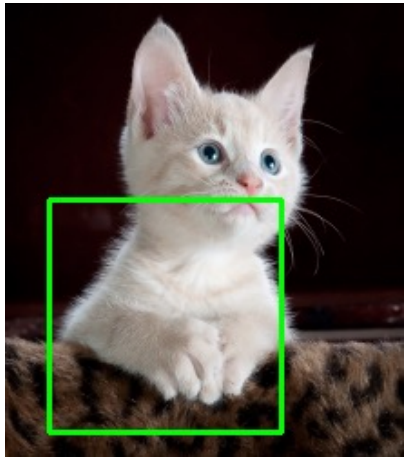
- Review: Tuples vs. Lists, 2D Lists
- IoU Calculation using Tuples
- NMS Using List and Tuples
- Review: Set and Dictionary
- One-Hot Encoding Using Set and Dictionary
- Review: Tree & K-DTree
- K-D Tree Implementation

AI VIET NAM
@aivietnam.edu.vn

Non-Maxima Suppression: List & Tuple







List

Tuple

```
# import the necessary packages
import numpy as np
import cv2
from google.colab.patches import cv2_imshow

boundingBoxes = np.array([
  (12, 84, 140, 212, 0.3),
  (24, 84, 152, 212, 0.4),
  (36, 84, 164, 212, 0.5),
  (12, 96, 140, 224, 0.6),
  (24, 96, 152, 224, 0.7),
  (24, 108, 152, 236, 0.8)])
```

Algorithm

Step 1 : Select the prediction **S** with highest confidence score and remove it from **P** and add it to the final prediction list **keep**. (**keep** is empty initially).

Step 2 : Now compare this prediction **S** with all the predictions present in **P**. Calculate the IoU of this prediction **S** with every other predictions in **P**. If the IoU is greater than the threshold **thresh_iou** for any prediction **T** present in **P**, remove prediction **T** from **P**.

Step 3 : If there are still predictions left in **P**, then go to **Step 1** again, else return the list **keep** containing the filtered predictions.

Q&A Time

Outline

- Review: Tuples vs. Lists, 2D Lists
- IoU Calculation using Tuples
- NMS Using List and Tuples
- Review: Set and Dictionary
- One-Hot Encoding Using Set and Dictionary
- Review: Tree & K-DTree
- K-D Tree Implementation

Set Review

❖ Create a set

Using curly brackets

```
1  # create a set
2  animals = {"cat", "dog", "tiger"}
3
4  print(type(animals))
5  print(animals)
```

```
<class 'set'>
{'dog', 'cat', 'tiger'}
```

Items with different data types

```
1  # create a set
2  a_set = {"cat", 5, True, 40.0}
3
4  print(type(a_set))
5  print(a_set)
```

```
<class 'set'>
{40.0, 'cat', 5, True}
```

Set comprehension

```
1  # set comprehension
2
3  a_set = {i*i for i in range(10)}
4  print(a_set)
```

```
{0, 1, 64, 4, 36, 9, 16, 49, 81, 25}
```

Dictionary Review

number_list = [11, 15, 35, 20, 18, 9]

11	15	35	20	18	9
0	1	2	3	4	5

Index

index

value

number_list[0] => 11

It's very hard to remember that the 1st element in a list has index 0, is there an easier way to remember this?

number_dict ={"One":11, "Two":15, "Three":35, "Four":20, "Five":18, "Six:9" }

11	15	35	20	18	9
"One	Two	Three	Four	Five	Six

key

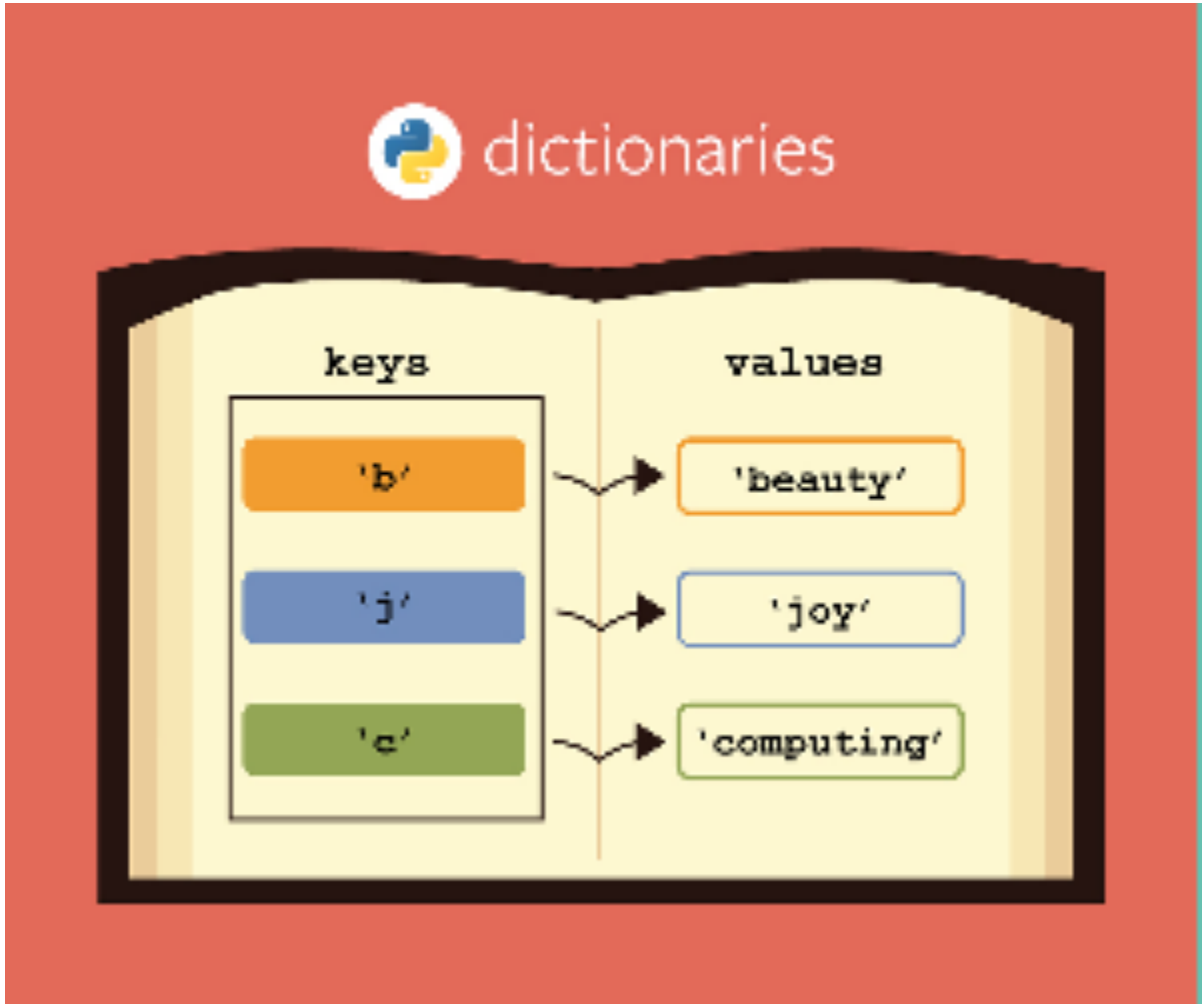
key

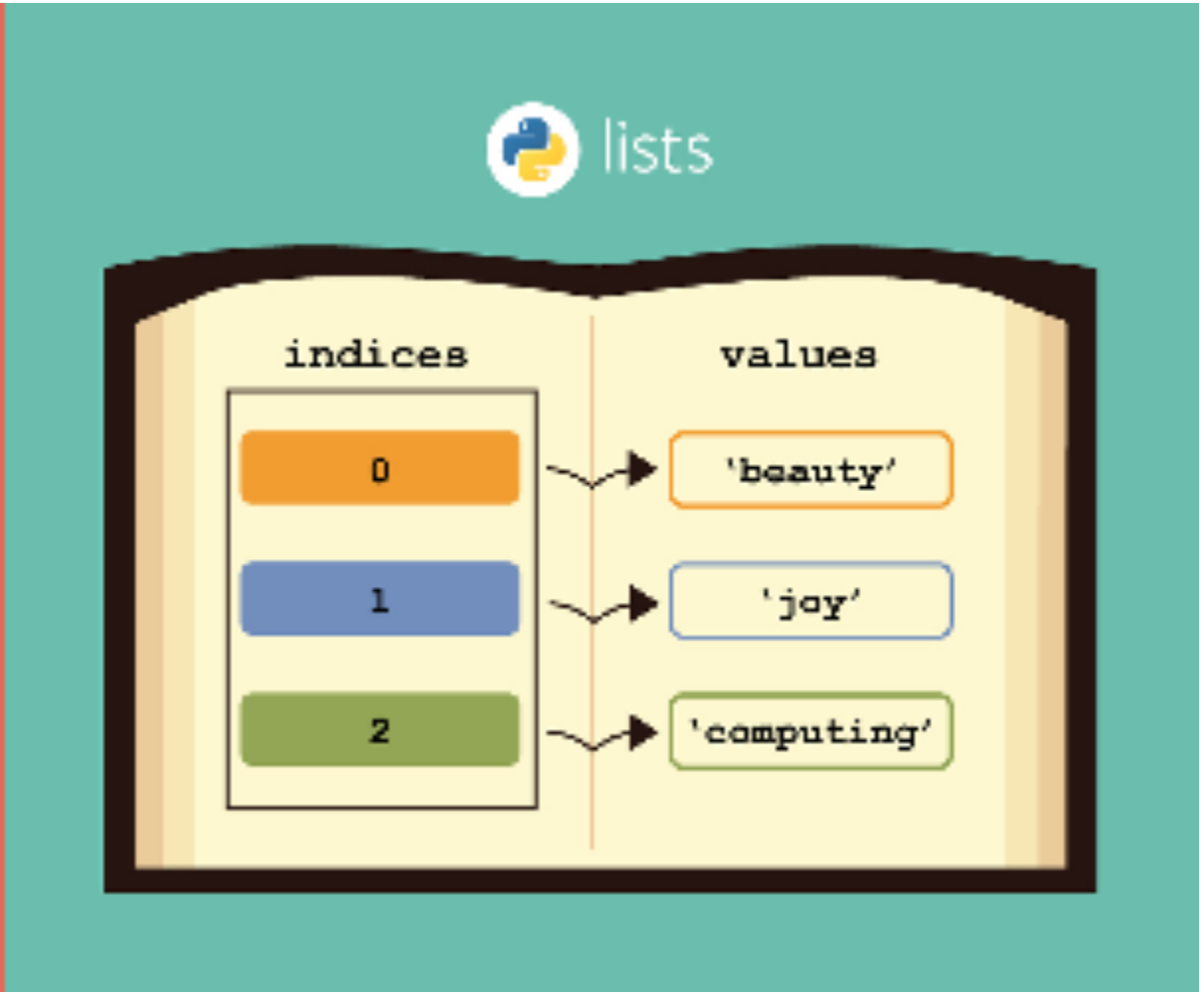
value

number_dict["One"] => 11

AI VIET NAM
@aivietnam.edu.vn

Dictionary Review

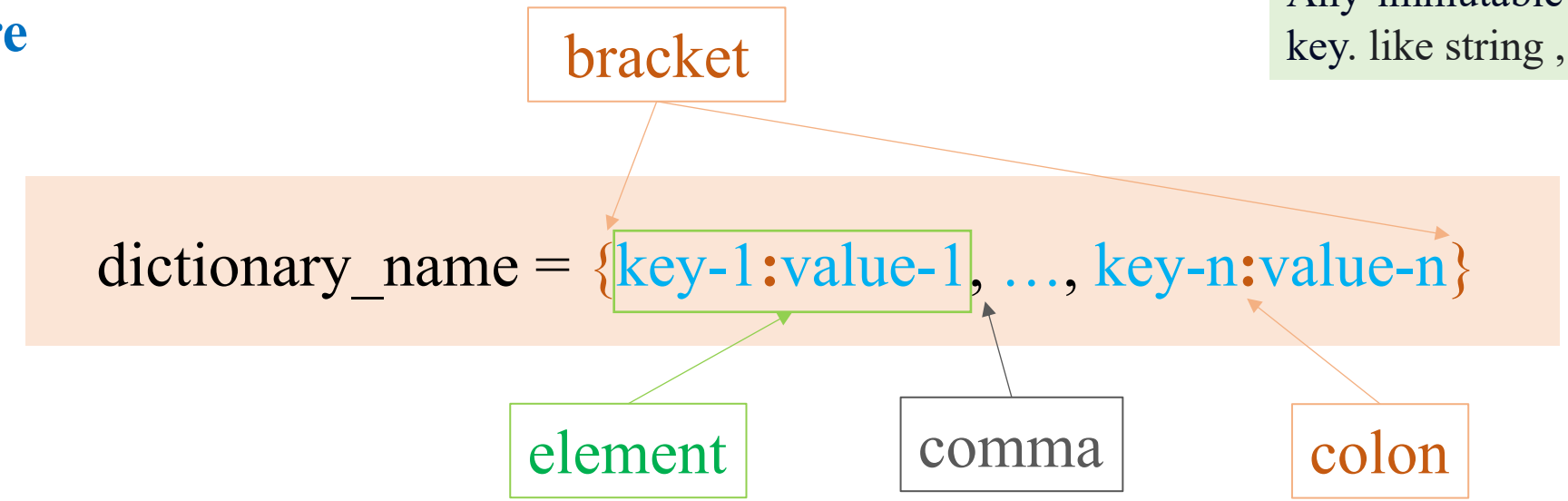




Dictionary Review

❖ Structure

Any immutable can be dictionary key. like string , number, tuple



Create a dictionary

```
1  parameters = {'learning_rate': 0.1,
2                'optimizer': 'Adam',
3                'metric': 'Accuracy'}
4
5  print(parameters)
6  print(type(parameters))
```

```
{'learning_rate': 0.1, 'optimizer': 'Adam', 'metric': 'Accuracy'}
<class 'dict'>
```

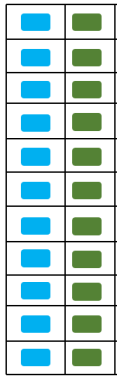
Outline

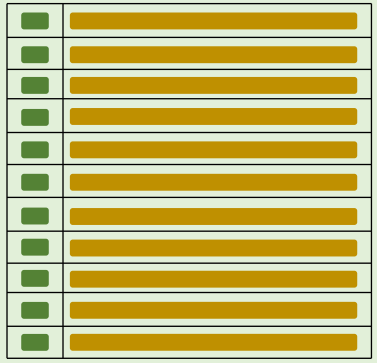
- Review: Tuples vs. Lists, 2D Lists
- IoU Calculation using Tuples
- NMS Using List and Tuples
- Review: Set and Dictionary
- One-Hot Encoding Using Set and Dictionary
- Review: Tree & K-DTree
- K-D Tree Implementation

AI VIET NAM
@aivietnam.edu.vn

Set/Dictionary in ML Preprocessing

❖ Text classification



Text Sample → Preprocessing (standardization & vectorization) →  → Machine Learning Algorithm → Output

”Make money” → Preprocessing → Embedding → Fully-connected layer

0	1	2	4	5	6	7	4	8

Embedding features → Support Vector Machine → Spam

Set/Dictionary in ML Preprocessing

How to Convert Categorical Data to Numerical Data?

1.Integer Encoding (already discussed in previous modules)

2.One-Hot Encoding

❖Vocabulary Building

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented and functional programming.

Python Program (using Set structure)

0 its	8 it	16 functional	24 design
1 indentation	9 including	17 readability	25 a
2 collected	10 and	18 general	26 the
3 supports	11 typed	19 with	27 oriented
4 is	12 interpreted	20 philosophy	28 emphasizes
5 programming	13 dynamically	21 object	29 high
6 garbage	14 multiple	22 level	30 significant
7 purpose	15 python	23 paradigms	31 structured
			32 language
			33 code
			34 use
			35 of

Vocabulary

Input

"Python is a high-level"

Text data

Output

15	4	25	29	22

Numerical Data

Set/Dictionary in ML Preprocessing

How to Convert Categorical Data to Numerical Data?

1.Integer Encoding (already discussed in previous modules)

2.One-Hot Encoding

Name
Ho Chi Minh
Da Nang
Ho Chi Minh
Da Nang
Can Tho

Integer Encoding
0
1
0
1
2

Integer Encoding

Can Tho	Da Nang	Ho Chi Minh
0	0	1
0	1	0
0	0	1
0	1	0
1	0	0

One-Hot Encoding

AI VIET NAM
@aivietnam.edu.vn

Set/Dictionary in ML Preprocessing

How to Convert Categorical Data to Numeri

1.Integer Encoding (already discussed in pre

2.One-Hot Encoding

Can Tho	Da Nang	Ho Chi Minh
0	0	1
0	1	0
0	0	1
0	1	0
1	0	0

One-Hot Encoding

```
One-Hot Encoded Features:
 [[0, 0, 1], [0, 1, 0], [0, 0, 1], [0, 1, 0], [1, 0, 0]]
['Can Tho', 'Da Nang', 'Ho Chi Minh']
```

```python
import numpy as np

dataset = np.array(['Ho Chi Minh', 'Da Nang',
                    'Ho Chi Minh', 'Da Nang','Can Tho'])

# Initialize an empty set to store unique categories
unique_categories = set()

# Iterate through the dataset and add categories to the set
for data in dataset:
    unique_categories.add(data)

# Convert the set to a sorted list to maintain order
unique_categories = sorted(list(unique_categories))

# Create a mapping from category to index
category_to_index = {category: idx for idx, category in enumerate(unique_categories)}

# Initialize an empty list to store the one-hot encoded features
one_hot_encoded_features = []

# Perform one-hot encoding
for data in dataset:
    one_hot_vector = [0] * len(unique_categories)
    one_hot_vector[category_to_index[data]] = 1
    one_hot_encoded_features.append(one_hot_vector)

# Print the unique categories and one-hot encoded features
print("One-Hot Encoded Features:\n", one_hot_encoded_features)
print(unique_categories)
```

Q&A Time

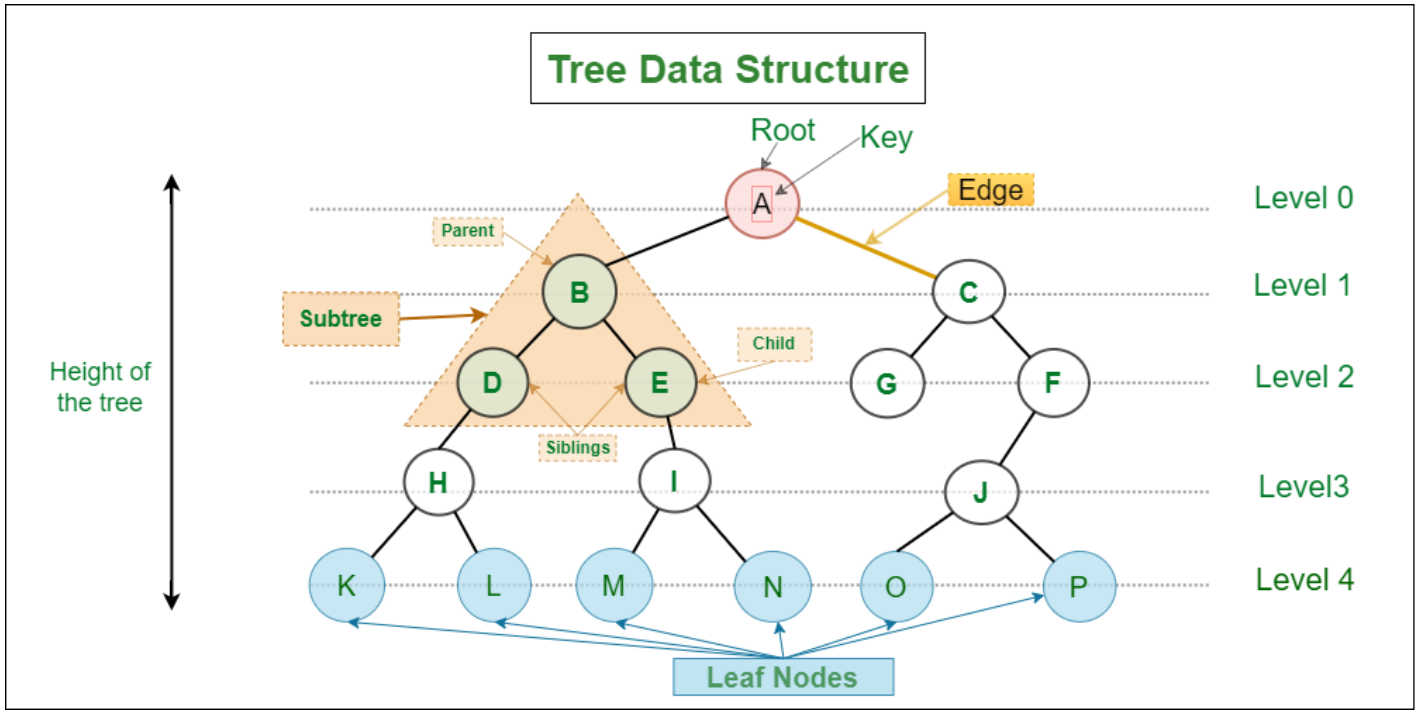
Outline

- Review: Tuples vs. Lists, 2D Lists
- IoU Calculation using Tuples
- NMS Using List and Tuples
- Review: Set and Dictionary
- One-Hot Encoding Using Set and Dictionary
- Review: Tree & K-DTree
- K-D Tree Implementation

AI VIET NAM
@aivietnam.edu.vn

Tree Review

A non-linear data structure where nodes are organized in a hierarchy



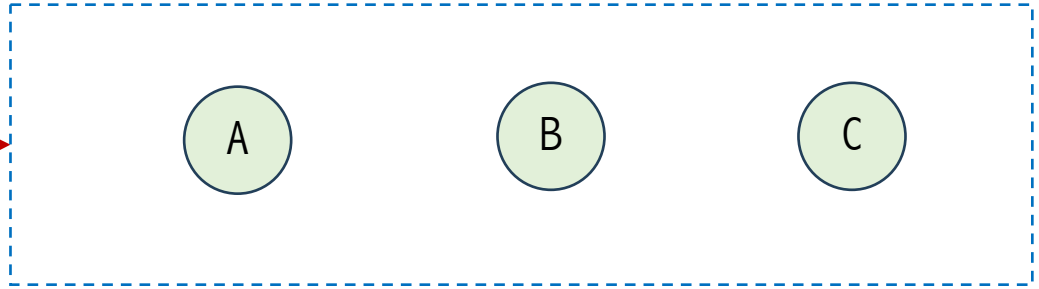
In the above figure, where do you think the root of the tree is?

Tree Review

```
class TreeNode:
    def __init__(self, data):
        self.data = data
        self.children = []
        self.parent = None
```

Node of a Tree

```
a_node = Node("A")
b_node = Node("B")
c_node = Node("C")
```



Create nodes, A, B, and C

Tree Review

```python
class TreeNode:
    def __init__(self, data):
        self.data = data
        self.children = []
        self.parent = None

    def add_child(self, child):
        child.parent = self
        self.children.append(child)

    def get_level(self): #to get level of each node
        level =0
        p = self.parent
        while p:
            level += 1
            p = p.parent

        return level
    def print_tree(self):
        space = ' ' * self.get_level() * 3
        prefix = space + '|__' if self.parent else ''
        print(prefix + self.data)#add prefix
        if self.children:
            for child in self.children:
                child.print_tree()
```

```python
def create_tree():
  a_node = TreeNode("A")
  b_node = TreeNode("B")
  c_node = TreeNode("C")
  d_node = TreeNode("D")
  e_node = TreeNode("E")
  f_node = TreeNode("F")
  g_node = TreeNode("G")

  a_node.add_child(b_node)
  a_node.add_child(c_node)

  b_node.add_child(d_node)
  b_node.add_child(e_node)

  c_node.add_child(f_node)
  c_node.add_child(g_node)
  return a_node
```

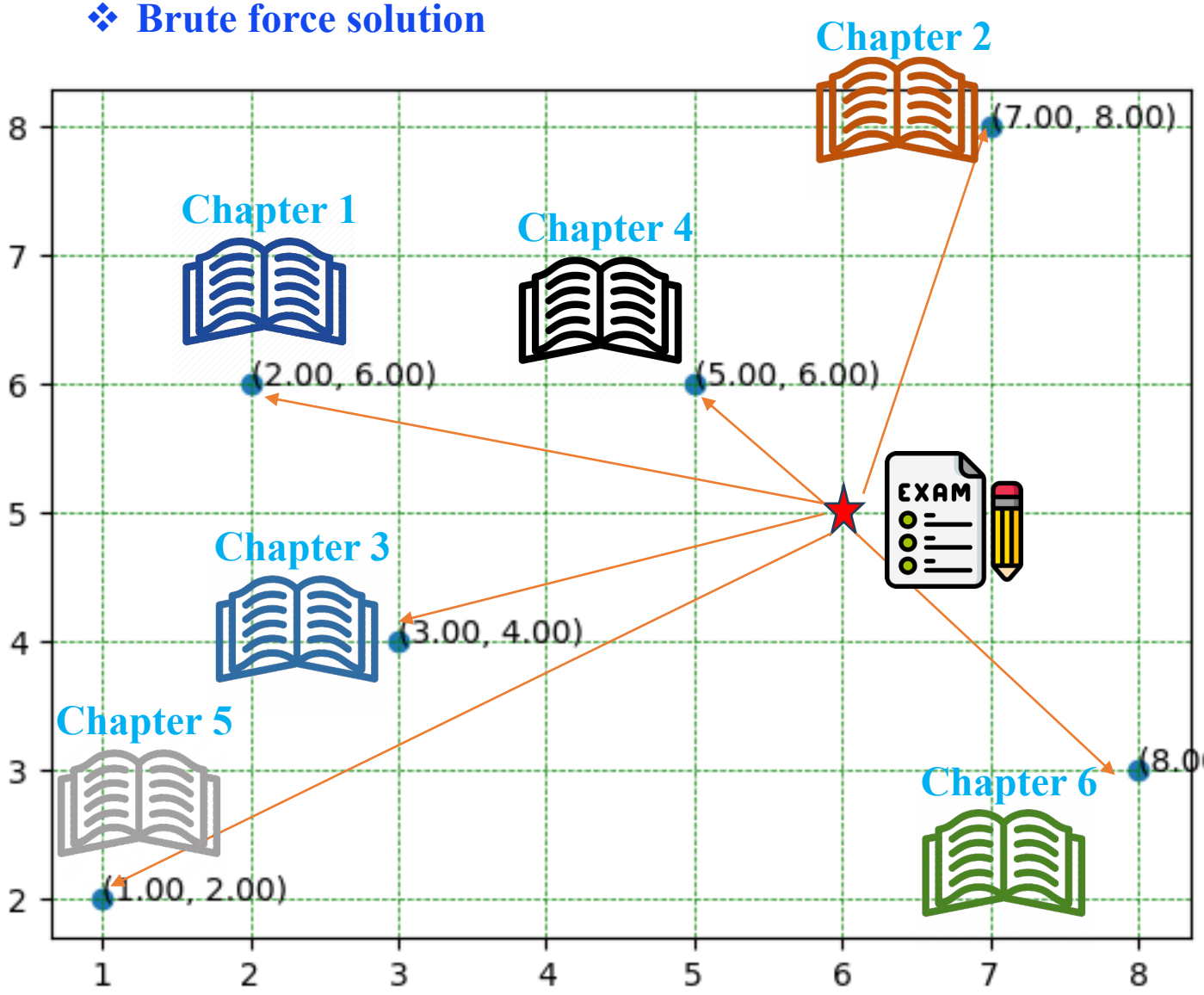
```python
tree = create_tree()
tree.print_tree()
```

```
A
    |__B
        |__D
        |__E
    |__C
        |__F
        |__G
```

AI VIET NAM
@aivietnam.edu.vn

Classification Problem

❖ **Brute force solution**




```
import matplotlib.pyplot as plt

x = [1, 2, 3, 5, 7, 8]
y = [2, 6, 4, 6, 8, 3]

for xy in zip(x, y):
    plt.annotate('(%.2f, %.2f)' % xy, xy=xy)

plt.scatter(x, y)
plt.grid(color = 'green', linestyle = '--', linewidth = 0.5)

plt.show()
```

★ **New datapoint**

1. Find distance of each point in the dataset
2. The point with lowest distance would be nears

K-D Tree Implementation Solution

Algorithm 1: K-D Tree Construction

Function *kdtree(pointList, depth)*:

Input: a list of points, *pointList*, and an integer, *depth* ;

Output: a k-d tree rooted at the median point of *pointList* ;

```
/* Select axis based on depth so that axis cycles
   through all valid values                          */
```

let $axis := depth \mod k$;

```
/* Sort point list and choose median as pivot element
   */
```

select median by *axis* from *pointList*;

```
/* Create node and construct subtree                 */
```

let *node.location* := *median*;

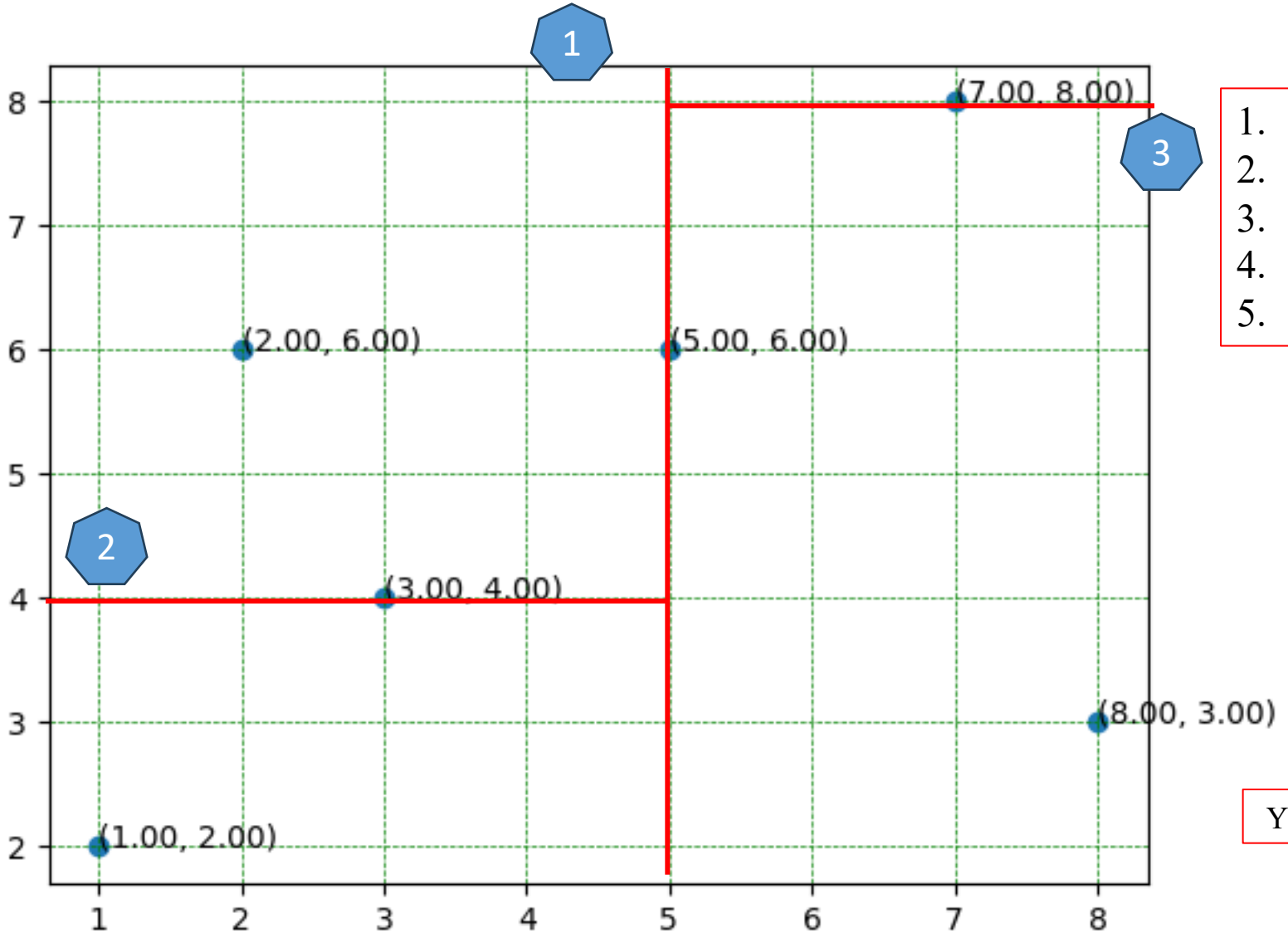
let *node.leftChild* := kdtree(points in *pointList* before *median*, $depth + 1$);

let *node.rightChild* := kdtree(points in *pointList* after *median*, $depth + 1$);

return *node*;

AI VIET NAM
@aivietnam.edu.vn

K-D Tree Implementation Solution



1. Pick any one feature at random
2. Find median
3. Split dataset in approximate equal halves
4. Pick next feature and repeat step #2,3
5. Continue until all data points are partitioned

X = 1, 2, 3, 5, 7, 8; median = 5

1

Left

Right

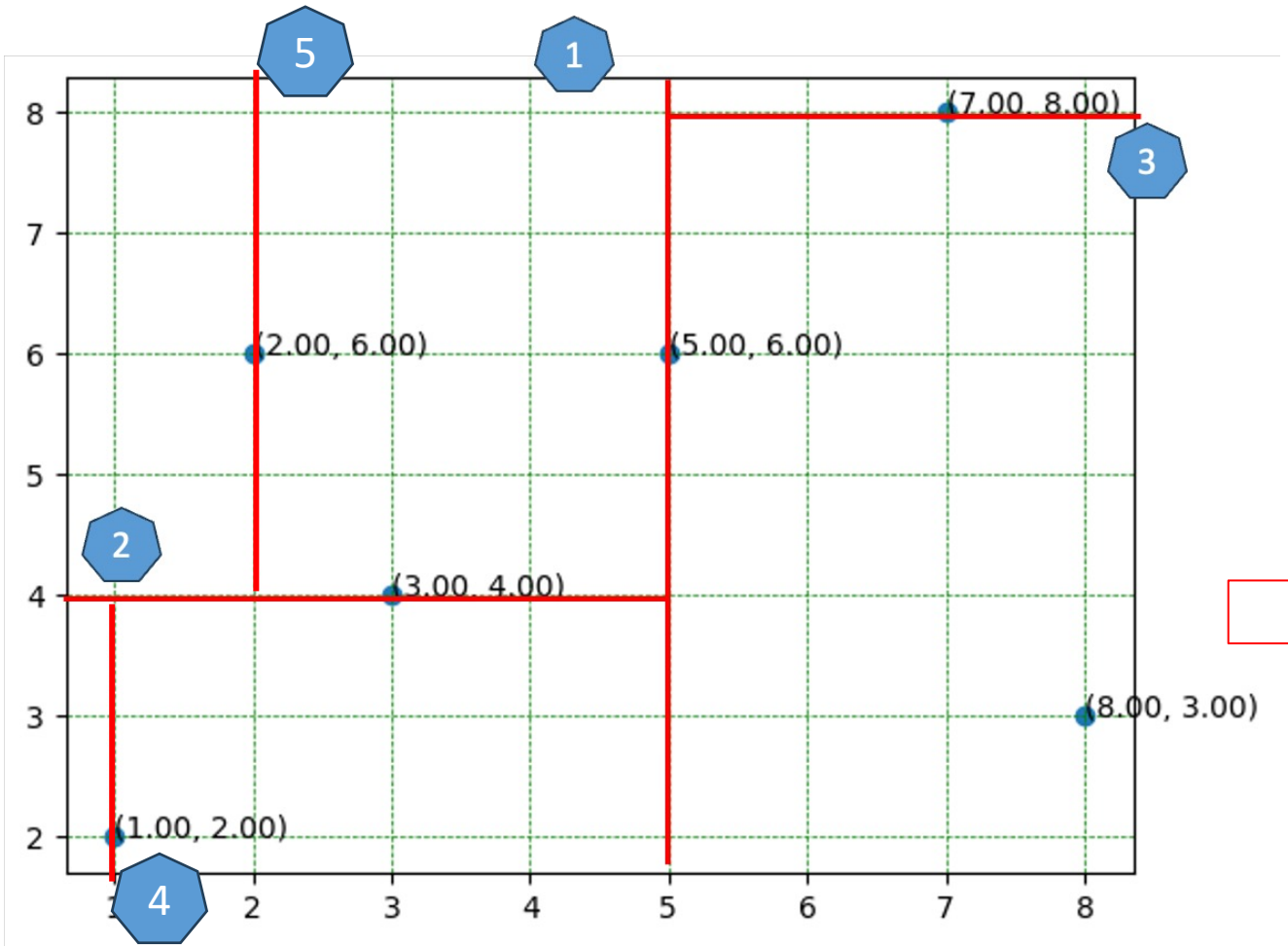
Y = 2, 4, 6; median = 4

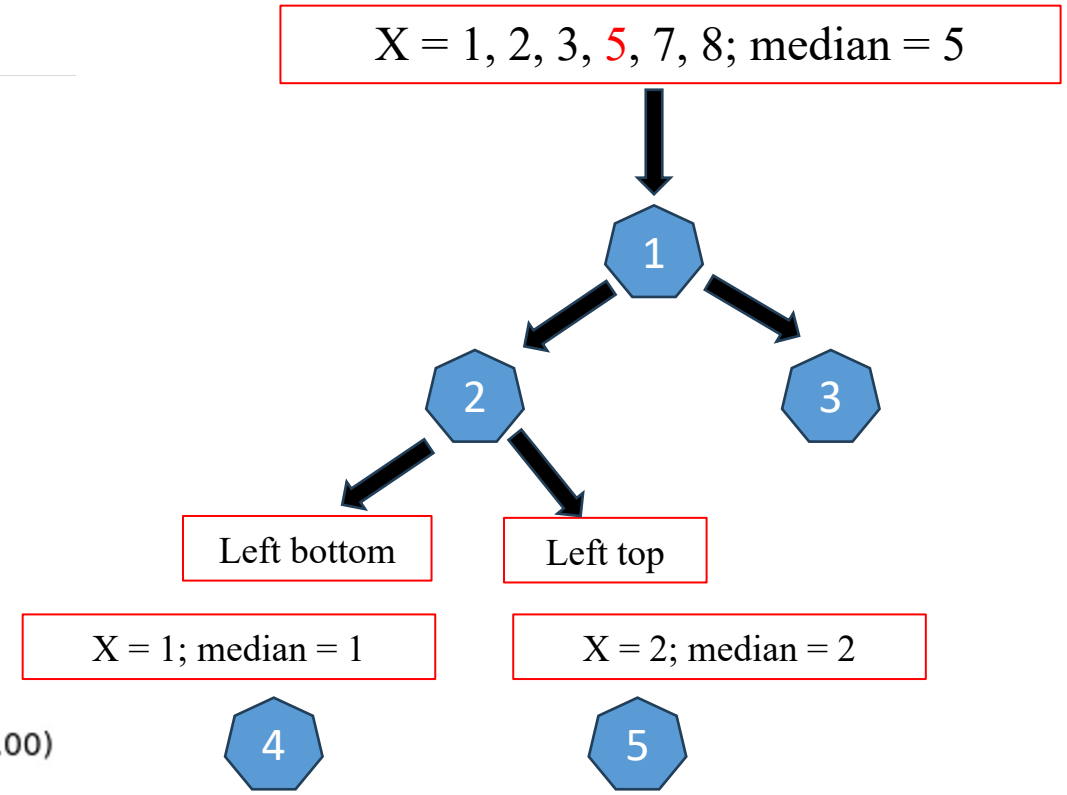
Y = 3, 8; median = 8

2

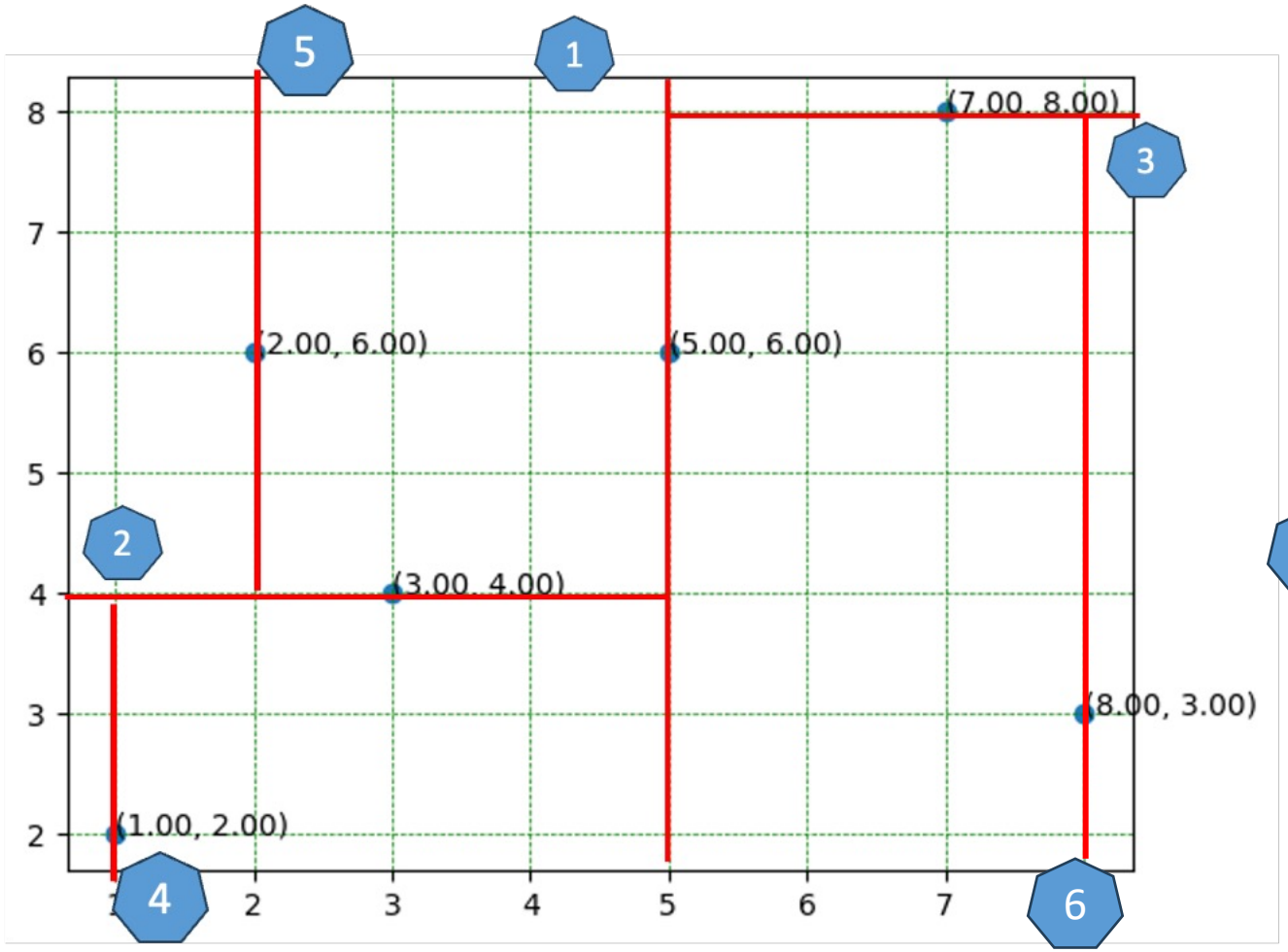
3

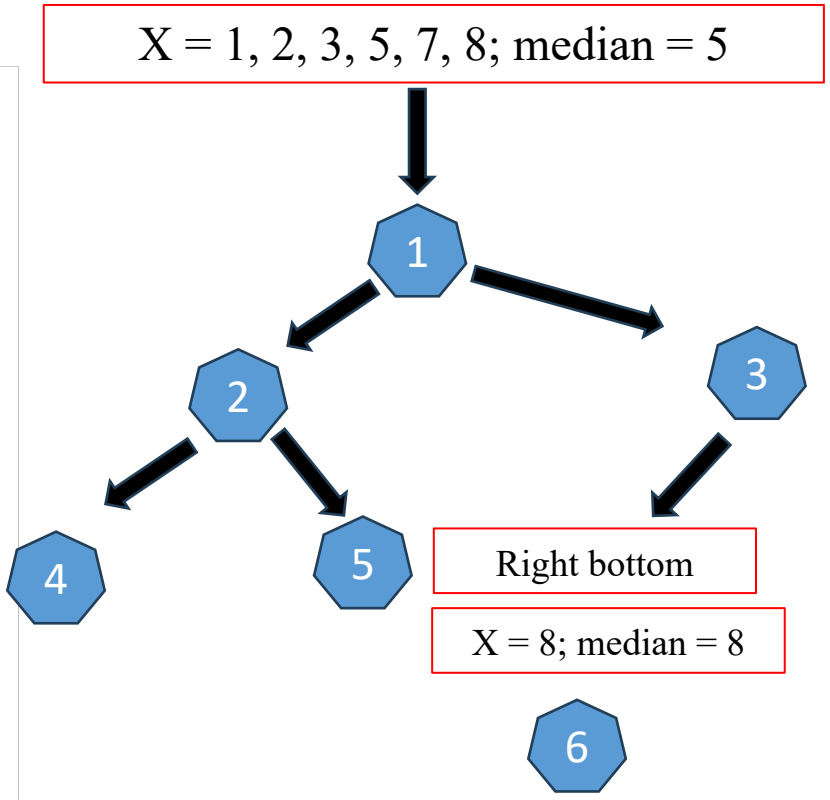
K-D Tree Implementation Solution



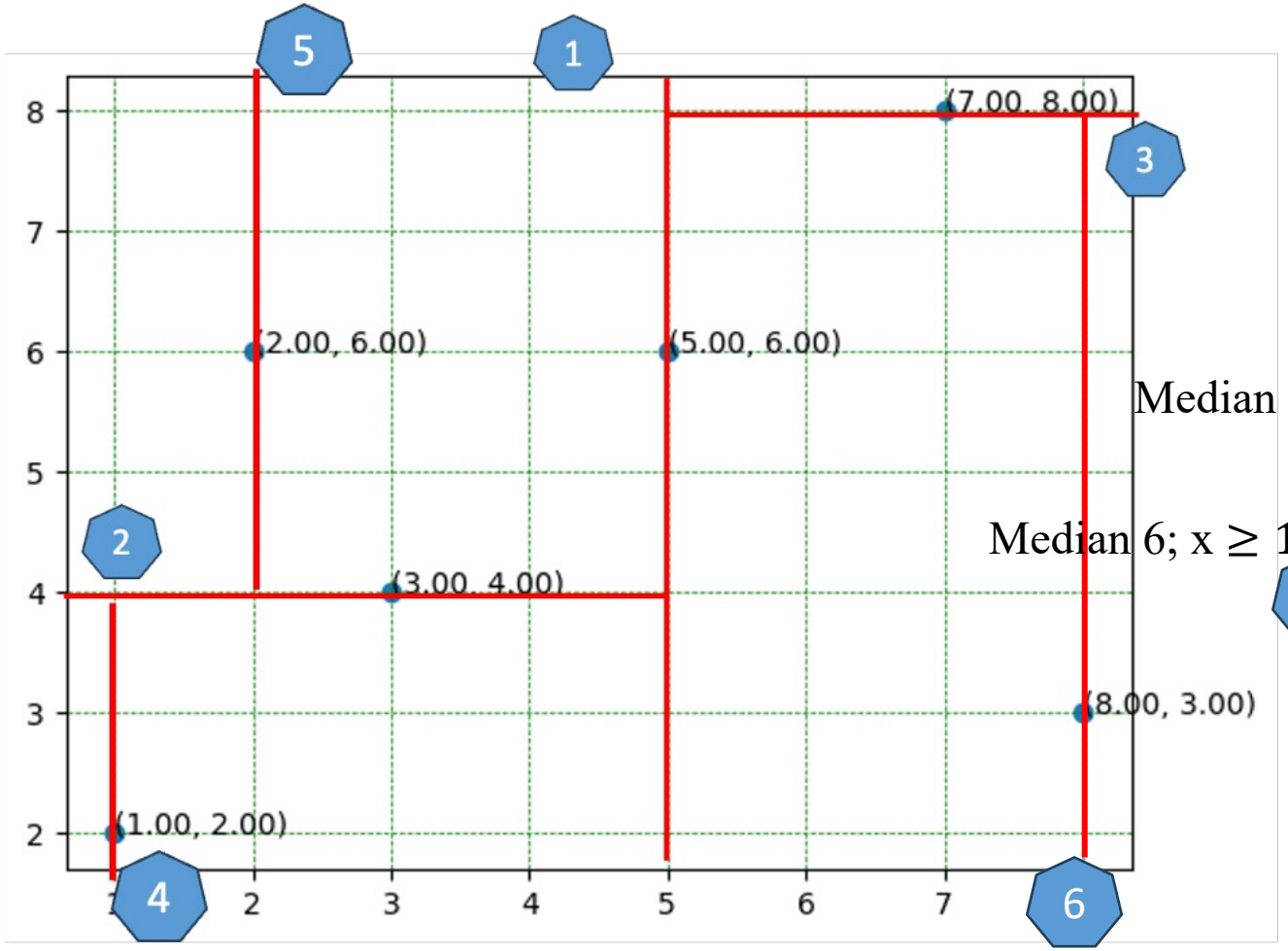


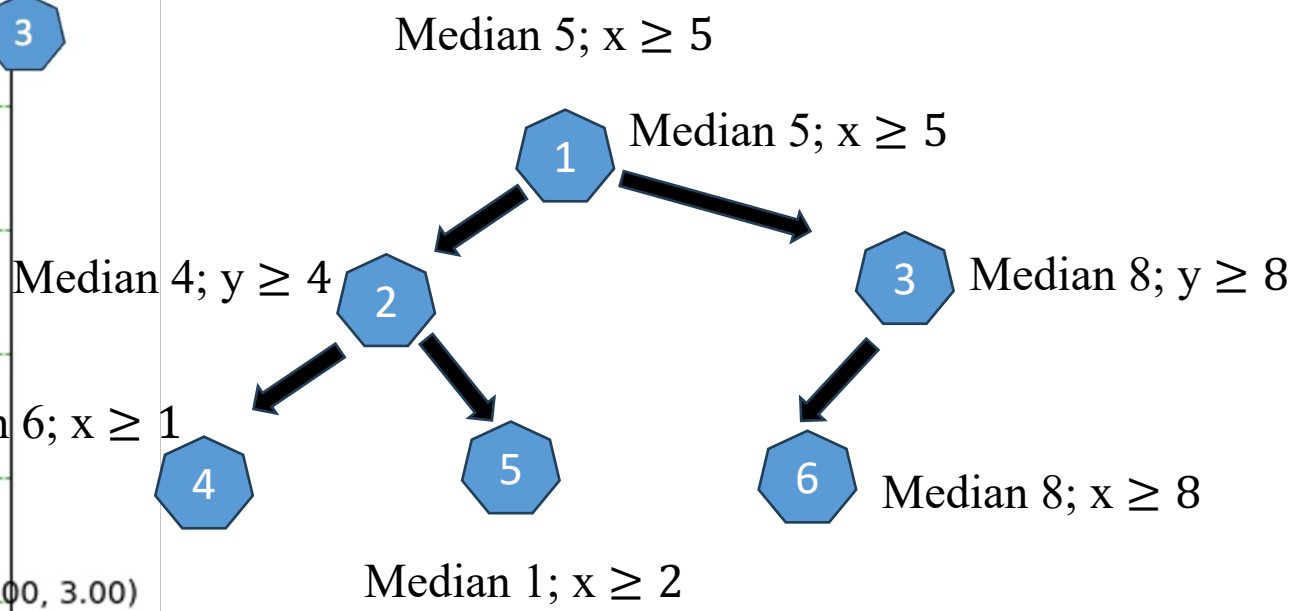
K-D Tree Implementation Solution





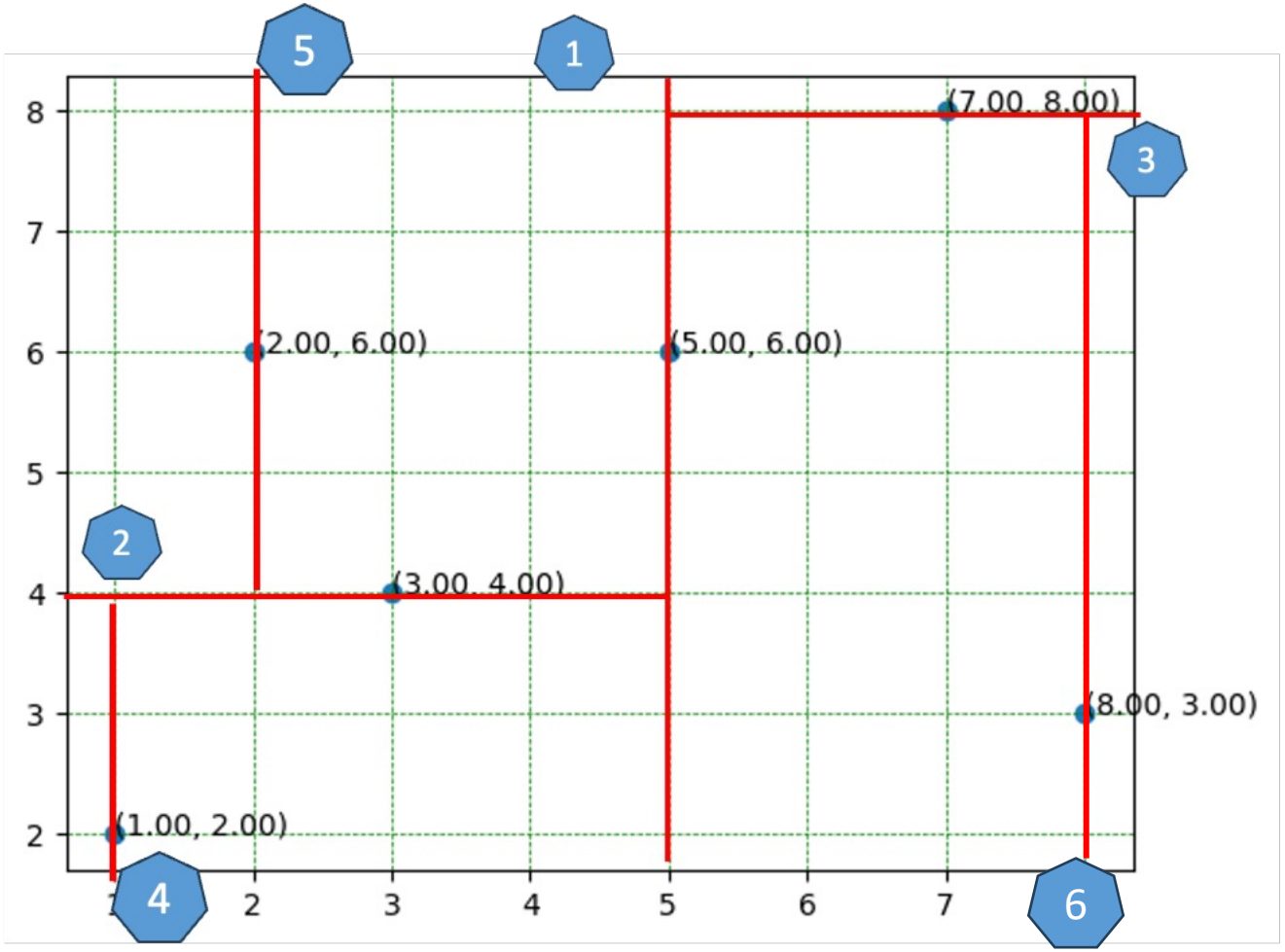
K-D Tree Implementation Solution

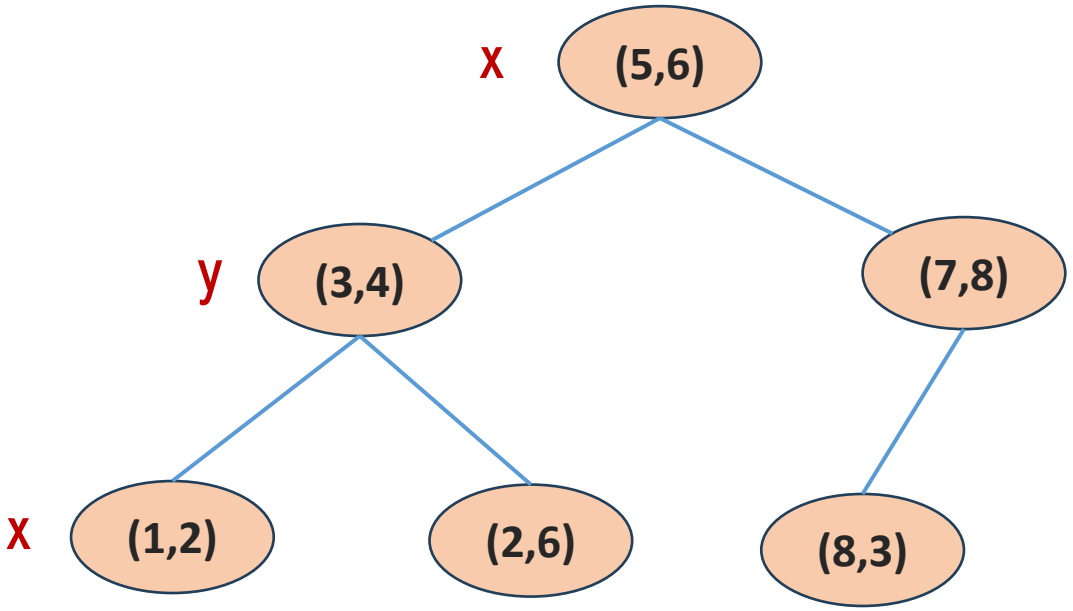




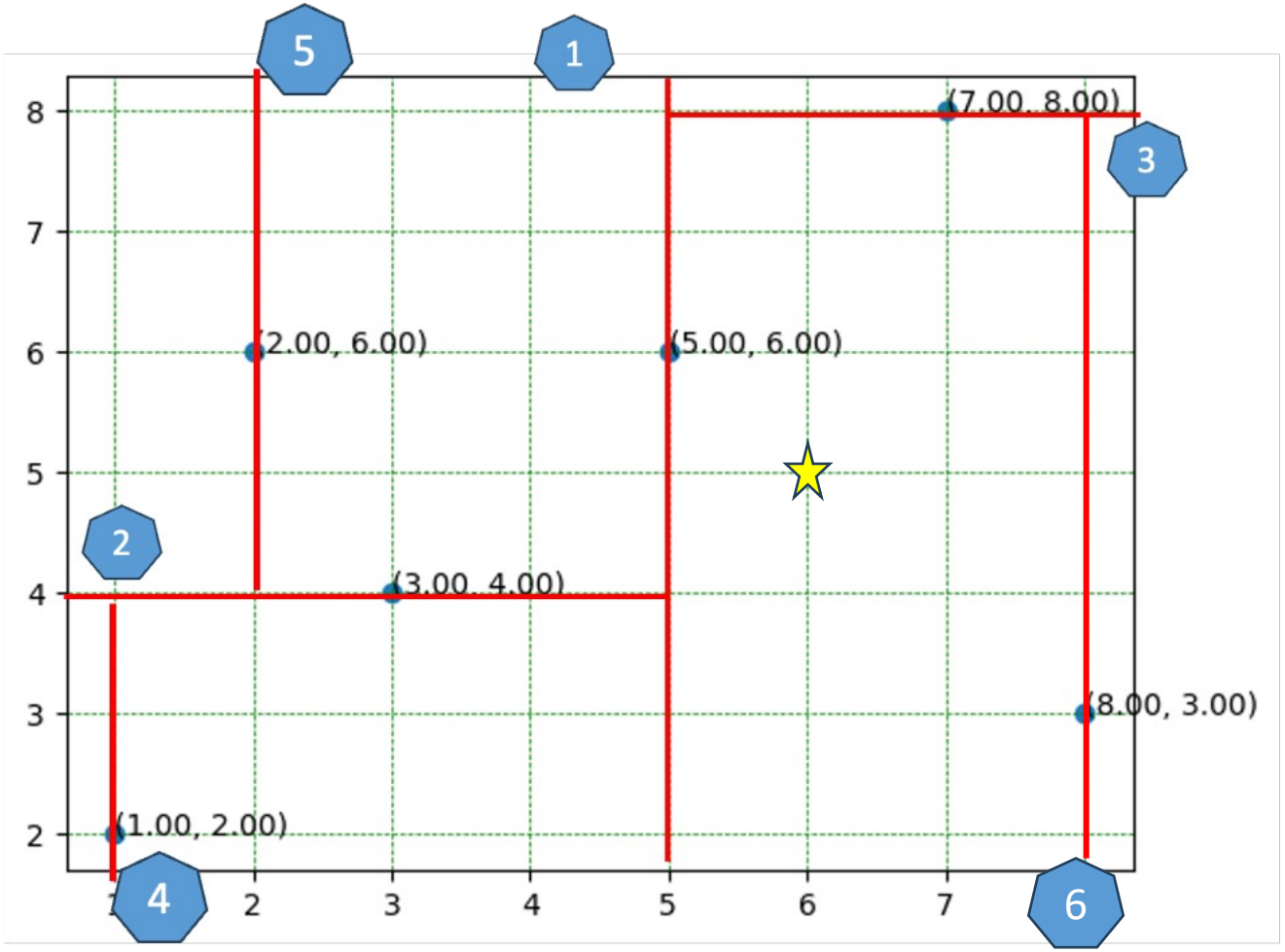
★ **New datapoint**

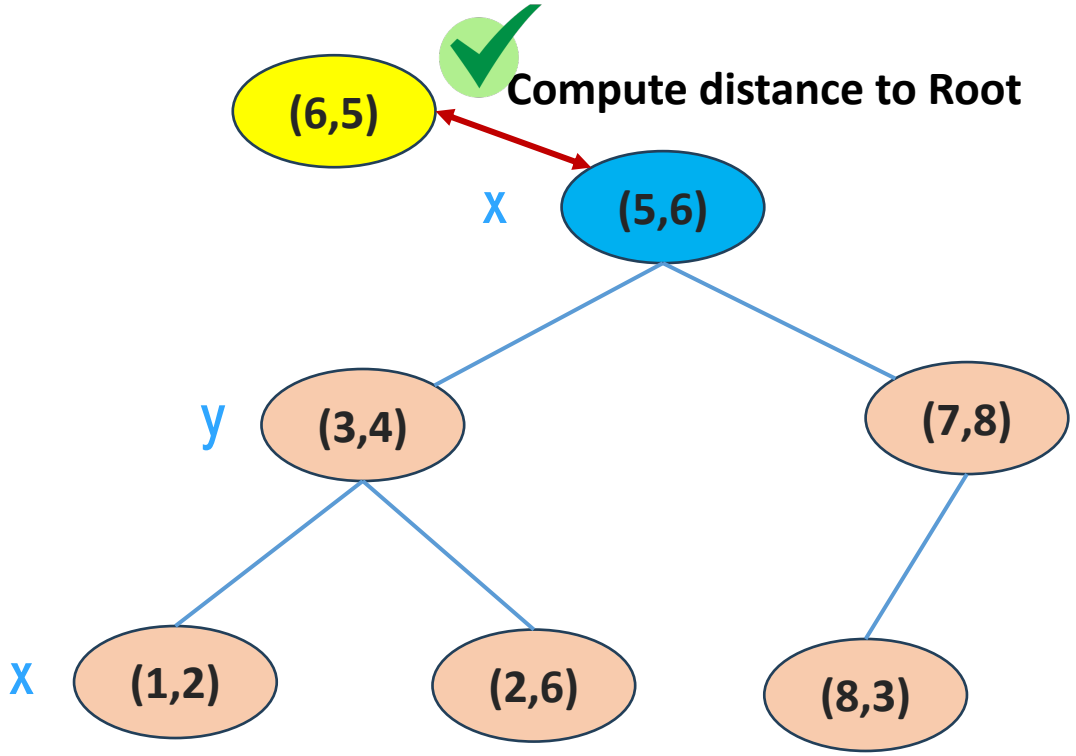
K-D Tree Implementation Solution





K-D Tree Implementation Solution

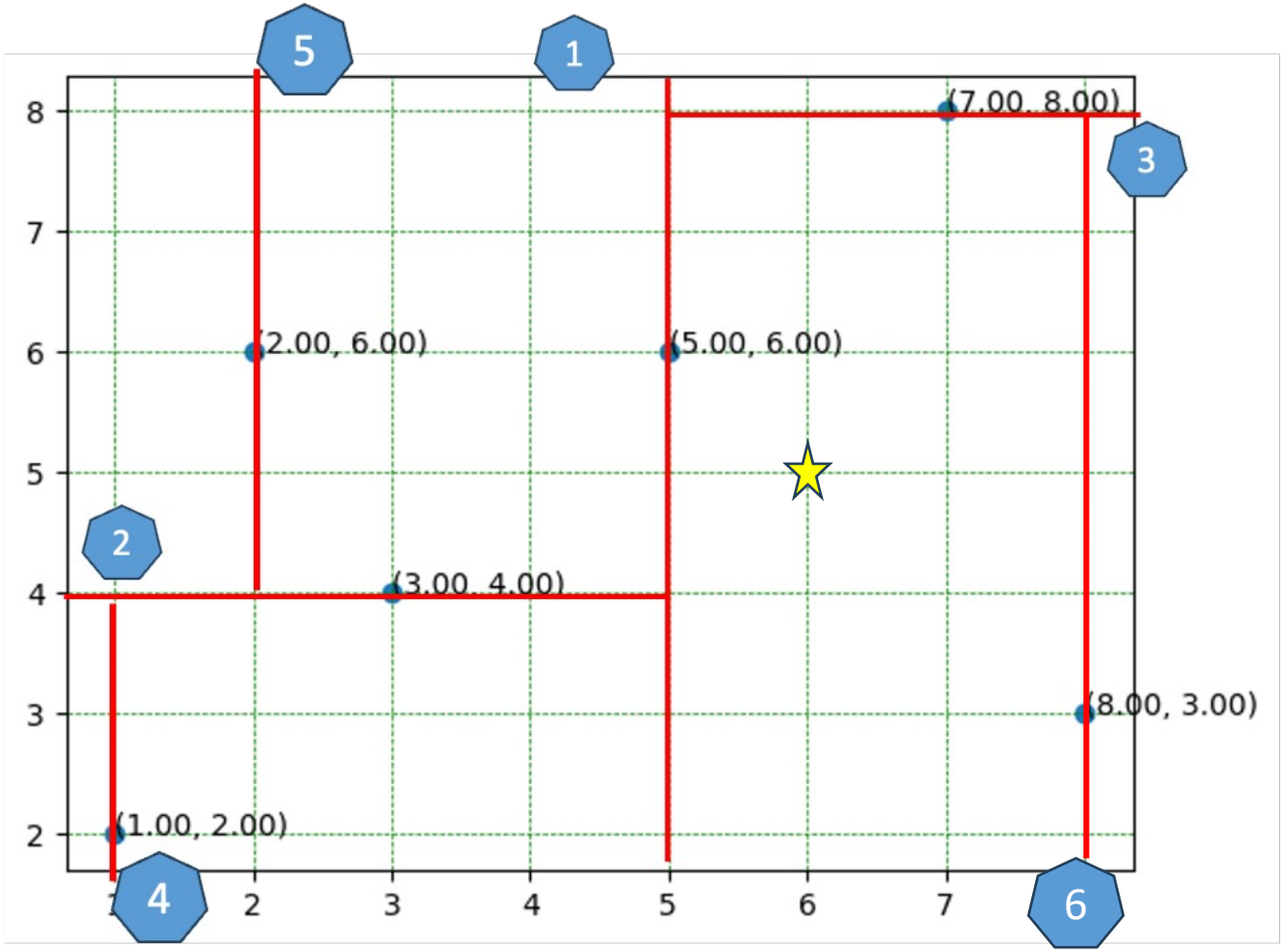


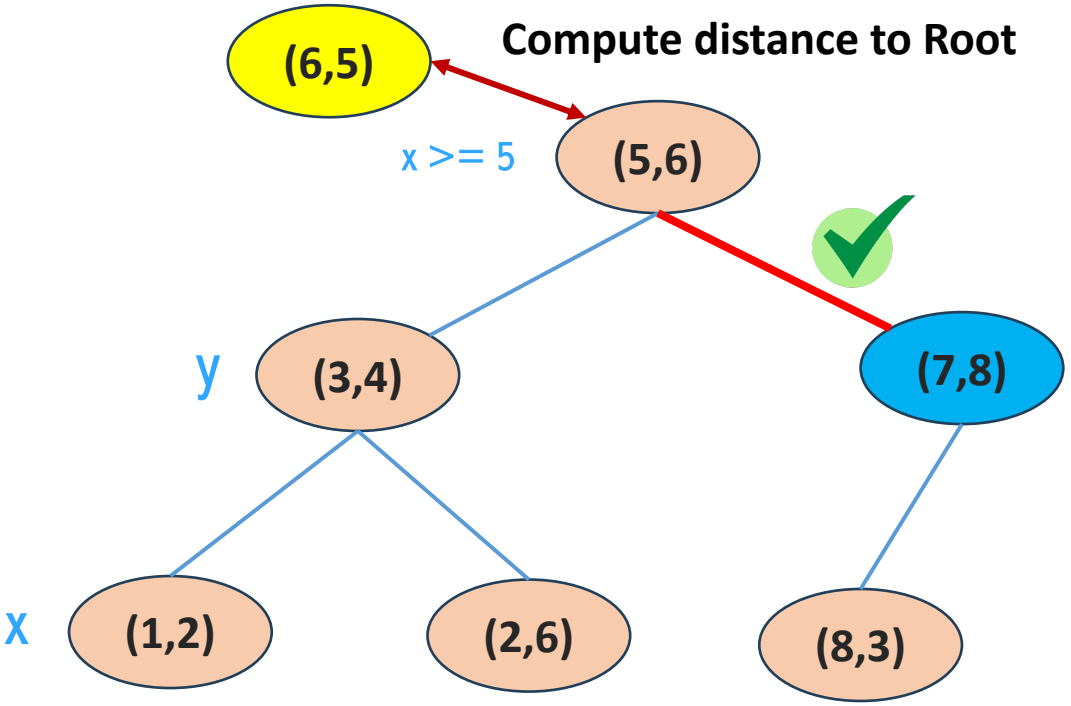


Best distance = 1.41
Best label = (5,6)

New datapoint

K-D Tree Implementation Solution





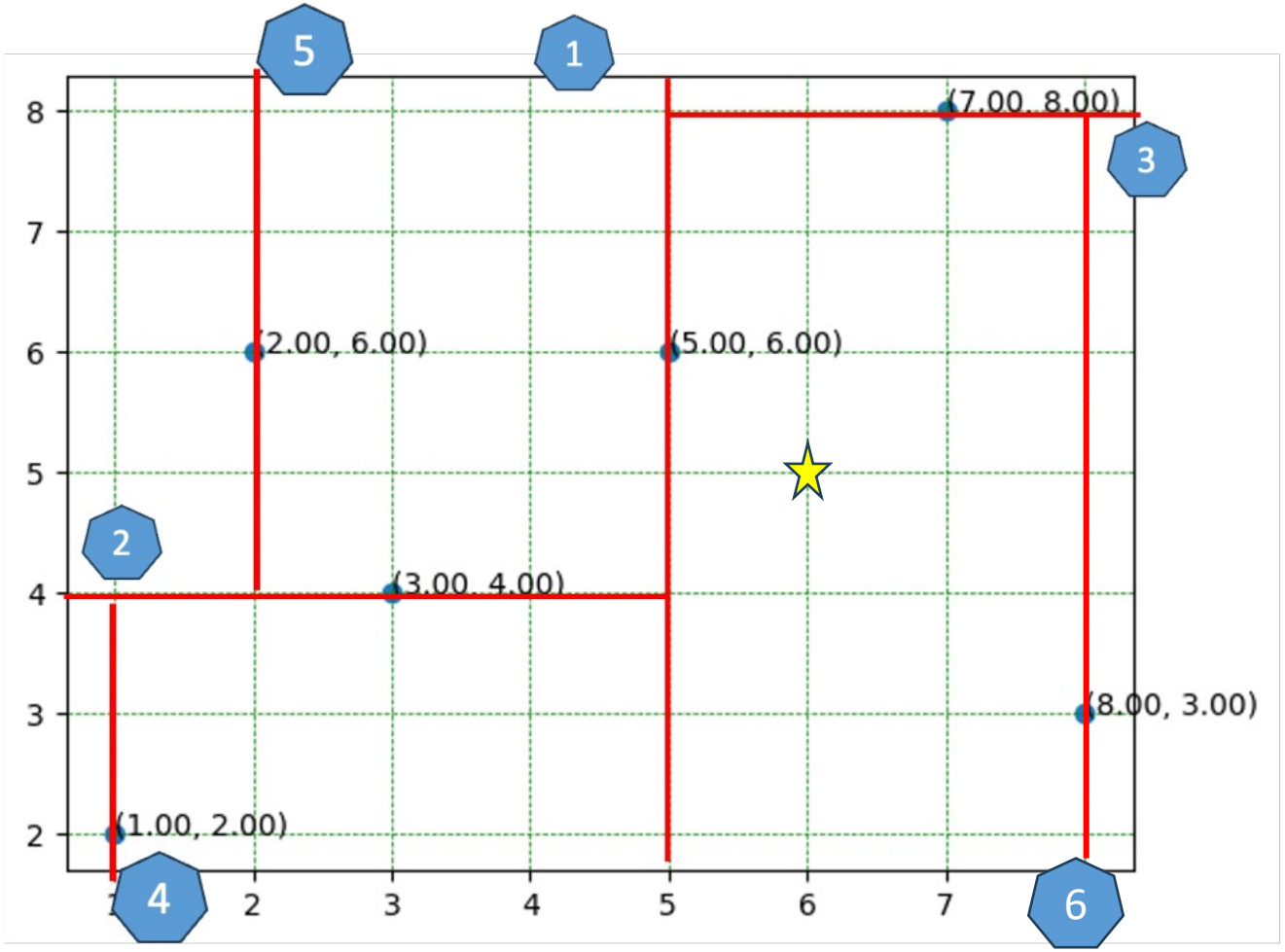
☆ **New datapoint**

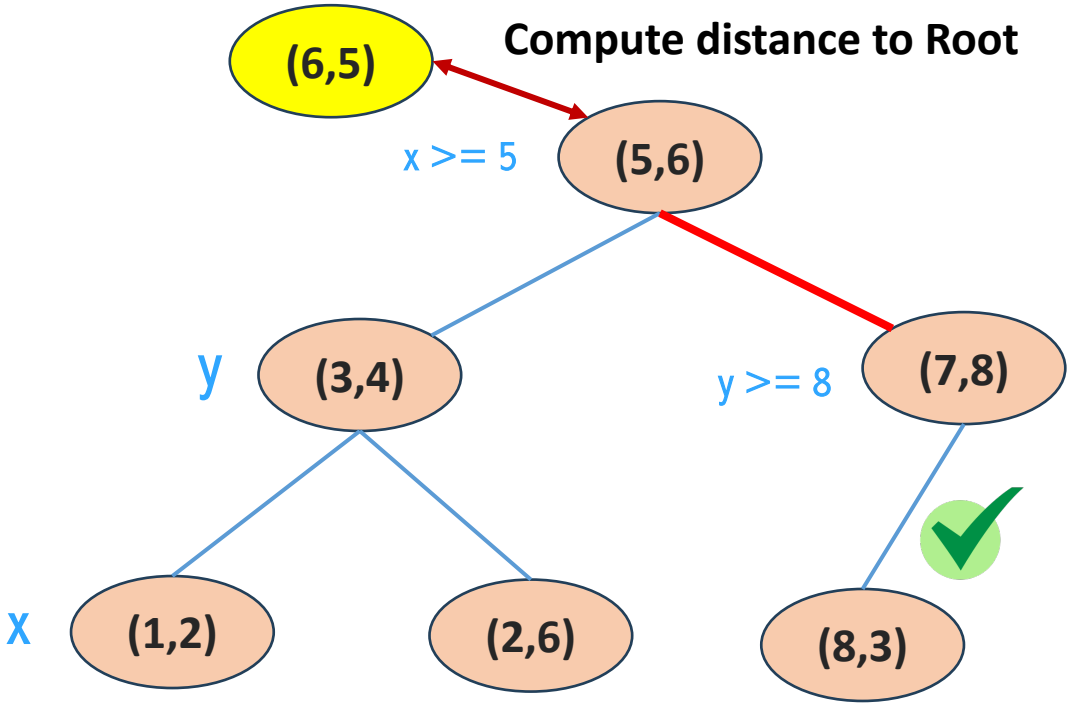
Best distance = 1.41
Best label = (5,6)

Curent distance = 3.16
Curent label = (5,6)

Not update the best distance

K-D Tree Implementation Solution





(6,5)

The closest points is (5,6)

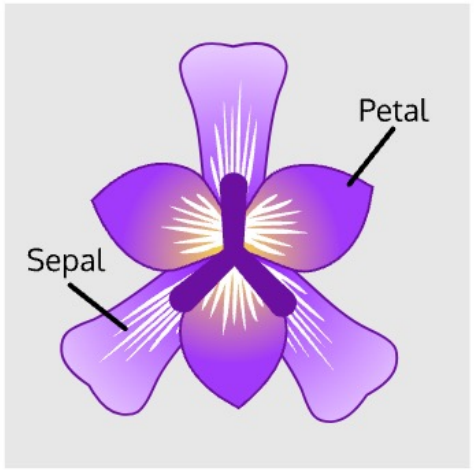
Best distance = 1.41
Best label = (5,6)

Curent distance = 2.82
Curent label = (8,3)

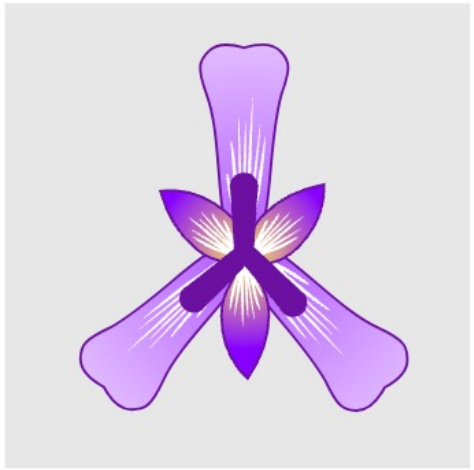
Not update the best distance

K-D Tree Implementation

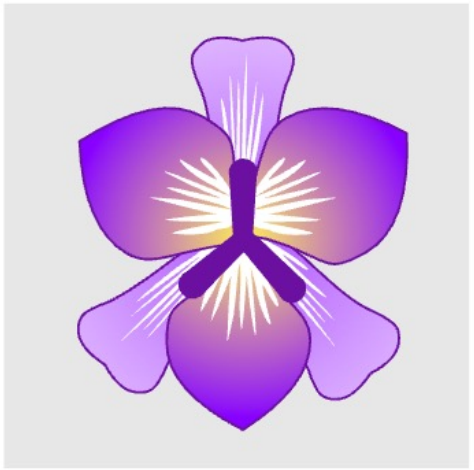
Iris Dataset Classification: Self-study



Iris Versicolor



Iris Setosa

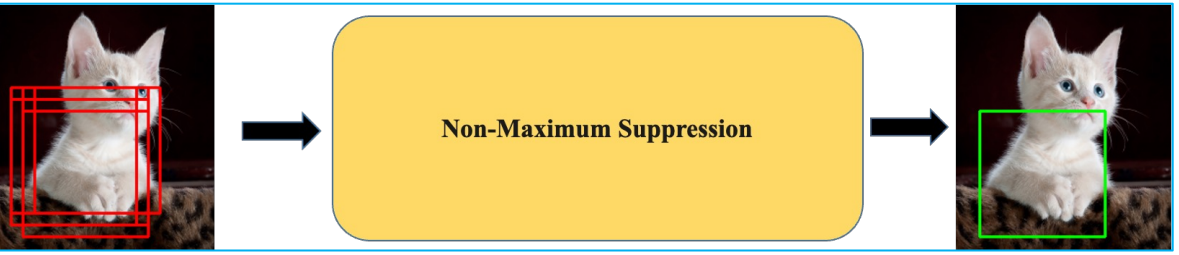


Iris Virginica

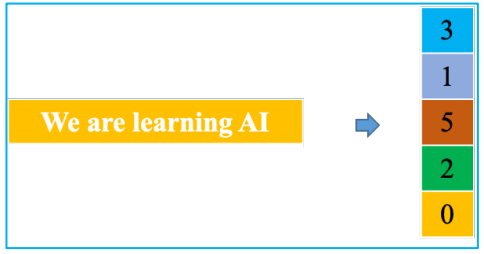
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Summary

Data Structure	Ordered	Mutable	Constructor	Example
List	Yes	Yes	[] or list()	[5. 7, ‘yes’, 5.7]
Tuple	Yes	No	() or tuple()	(5.7, ‘yes’, 5.7)
Set	No	Yes	{} or set()	{5.7, ‘yes’ }
Dictionary	No	Yes	{} or dict{}	{‘key’: value}



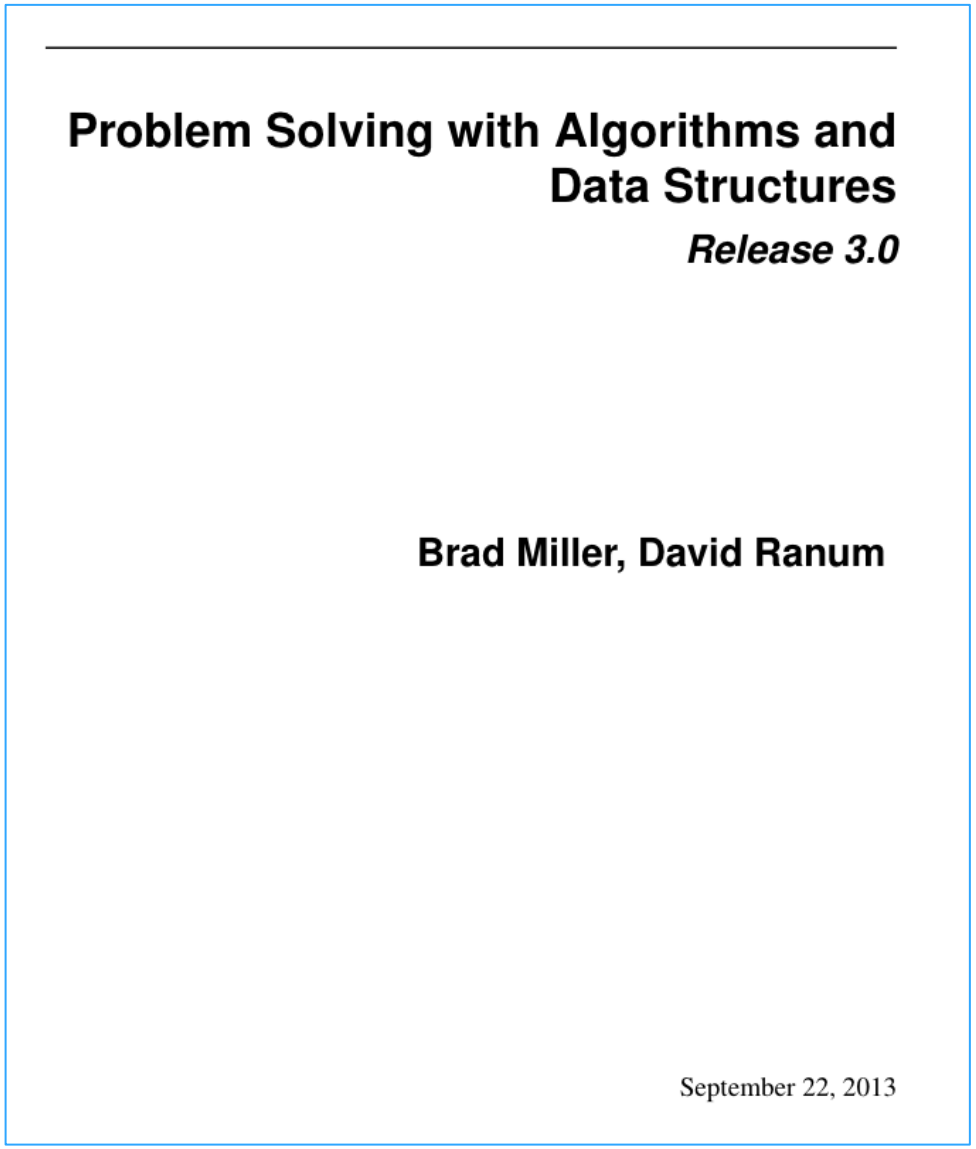
IOU, NMS using Tuple

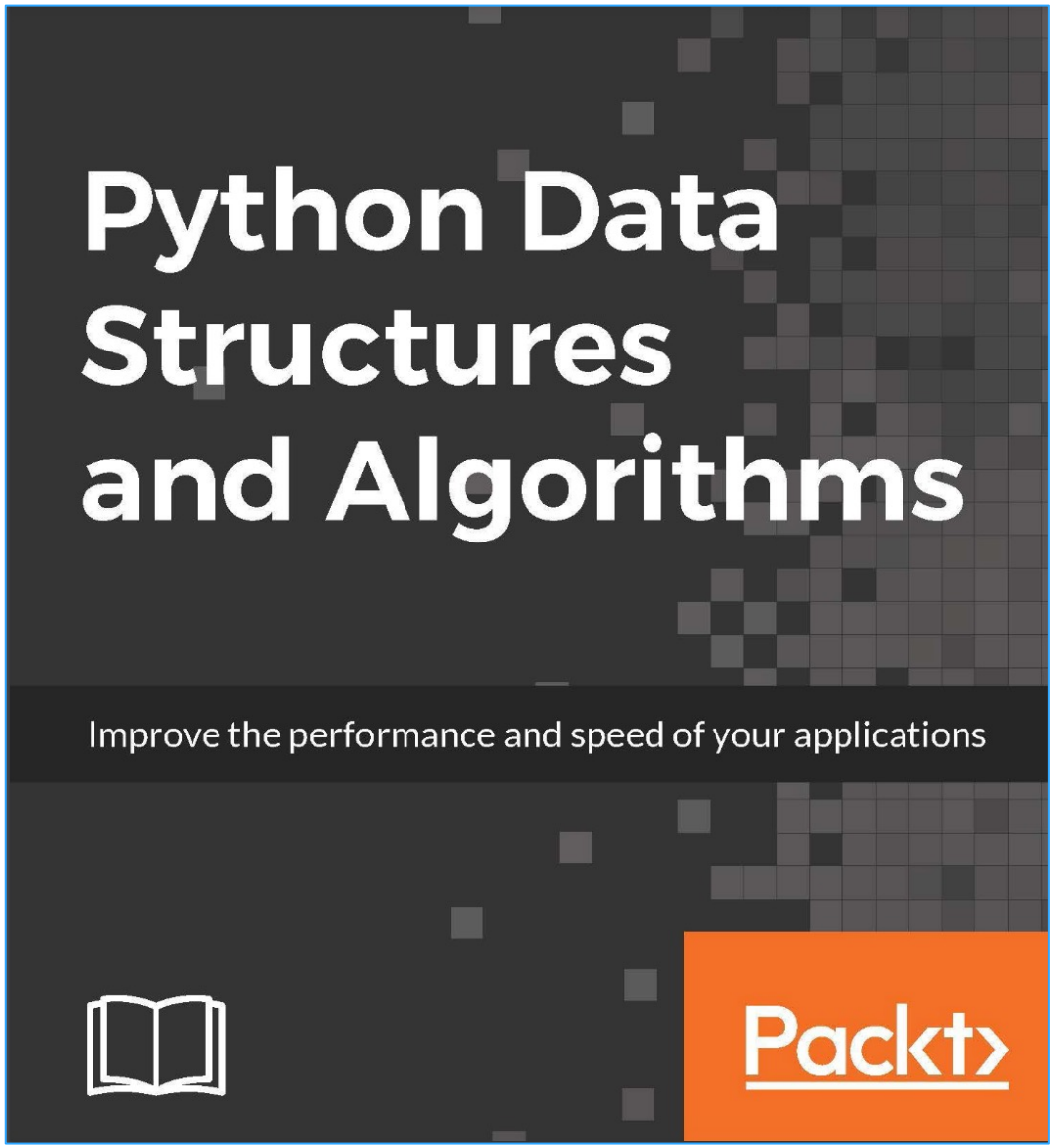


Text embedding using Set

Histogram using Dictionary

References


Problem Solving with Algorithms and Data Structures
Release 3.0
Brad Miller, David Ranum
September 22, 2013


Python Data Structures and Algorithms
Improve the performance and speed of your applications
Packt