TRÍ TUỆ NHÂN TẠO - ITEC3413 - 2025

LAB 02

```
import collections
import math
from typing import Any, DefaultDict, List, Set, Tuple
```

```
############################################################
def find_alphabetically_first_word(text:str)->str:
  '''
  Cho một chuỗi |text|, trả về từ trong |text| đứng đầu theo thứ tự từ điển
  (tức là từ sẽ đứng đầu sau khi sắp xếp). Một từ được định nghĩa bằng một tập
  các ký tự không có khoảng trắng. Nếu chuỗi đầu vào là một chuỗi rỗng, có thể
  trả về một chuỗi rỗng hoặc báo lỗi.
  Ví dụ:
  Input: "toi di hoc"
  Ouput: "di"

  Input: "toi thuc hanh tri tue nhan tao"
  Output: "hanh"
  '''
  # BEGIN_YOUR_CODE (our solution is 1 line of code, but don't worry if you deviate from this)
  raise Exception("Not implemented yet")
  # END_YOUR_CODE
```

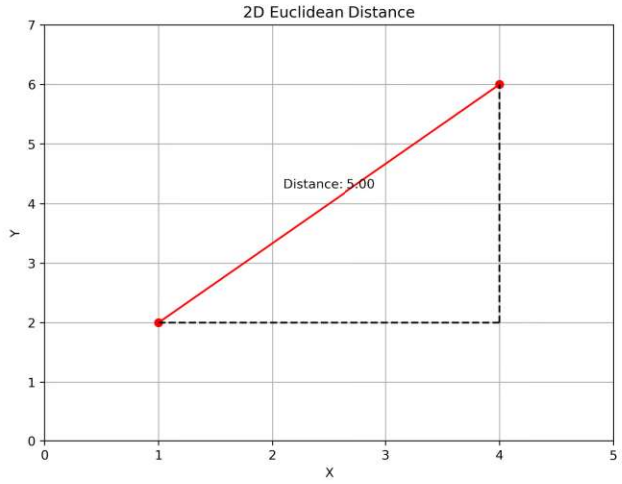
```
print(find_alphabetically_first_word("toi thuc hanh tri tue nhan tao"))
```

Euclidean distance

2D Euclidean distance

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$\begin{aligned} d &= \sqrt{(4-1)^2 + (6-2)^2} \\ &= \sqrt{3^2 + 4^2} \\ &= \sqrt{9 + 16} \\ &= \sqrt{25} \\ &= 5 \end{aligned}$$




```
############################################################
def euclidean_distance(loc1: Position, loc2: Position) -> float:
    """
    Viết hàm trả về khoảng cách Euclidean giữa hai vị trí (loc1 và loc2), trong
    đó vị trí (locations) là một cặp số (ví dụ: (3, 5)).
    """
    # BEGIN_YOUR_CODE (our solution is 1 line of code, but don't worry if you deviate from this)
    raise Exception("Not implemented yet")
    # END_YOUR_CODE
```

```
loc1 = (1, 2)
loc2 = (4, 6)
print(euclidean_distance(loc1, loc2))
```

Manhattan Distance

For 2-dimensional vectors:

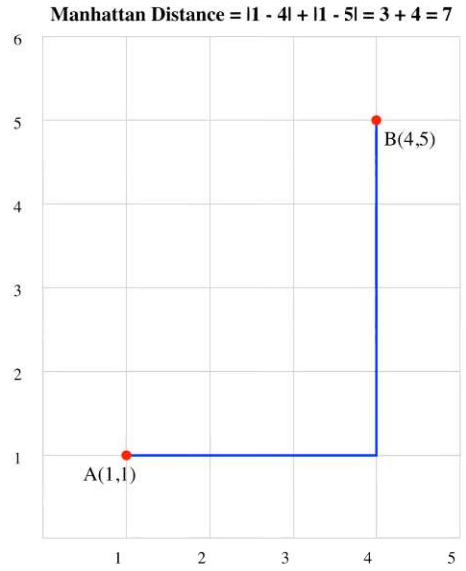
Manhattan distance = $|x_1 - x_2| + |y_1 - y_2|$

For 3-dimensional vectors:

Manhattan distance = $|x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2|$

General formula for n-dimensional vectors:

Manhattan distance = $|x_1 - x_2| + |y_1 - y_2| + \ldots + |v_{n1} - v_{n2}|$



```
############################################################
def manhattan_distance(loc1: Position, loc2: Position) -> float:
    """
    Viết hàm trả về khoảng cách Euclidean giữa hai vị trí (loc1 và loc2), trong
    đó vị trí (locations) là một cặp số (ví dụ: (3, 5)).
    """
    # BEGIN_YOUR_CODE (our solution is 1 line of code, but don't worry if you deviate from this)
    raise Exception("Not implemented yet")
    # END_YOUR_CODE
```

```
loc1 = (1, 1)
loc2 = (4, 5)
print(manhattan_distance(loc1, loc2))
```

```
############################################################

def sparse_vector_dot_product(v1: collections.defaultdict, v2: collections.defaultdict) -> float:
    """
    Cho hai vectors thưa (vectors mà hầu hết các phần tử là 0 - zeros): v1 và v2.
    Mỗi vector được biểu diễn bởi collections.defaultdict(float). Hãy tính tính
    vô hướng (dot product) của hai vectors đã cho.
    """
    # BEGIN_YOUR_CODE (our solution is 1 line of code, but don't worry if you deviate from this)
    raise Exception("Not implemented yet")
    # END_YOUR_CODE
```

```
v1 = collections.defaultdict(float, {'a':5, 'b':7})
v2 = collections.defaultdict(float, {'b':2, 'c': 4, 'd': 3})

print(sparse_vector_dot_product(v1, v2))
```

```
defaultdict(<class 'float'>, {})
```

```
############################################################

def increment_sparse_vector(v1: collections.defaultdict, scale: float,
                            v2: collections.defaultdict,
) -> None:
    """
    Cho hai vectors thưa: v1 và v2, thực hiện v1 += scale * v2.
    Nếu scale = 0 (zero),bạn được phép sửa đổi v1 để bao gồm các khóa bổ sung
    trong v2, hoặc không thêm khóa mới nào cả.
    """
    # BEGIN_YOUR_CODE (our solution is 2 lines of code, but don't worry if you deviate from this)
    raise Exception("Not implemented yet")
    # END_YOUR_CODE
```

```
v1 = collections.defaultdict(float, {'a': 5})
v2 = collections.defaultdict(float, {'b': 2, 'a': 3})

increment_sparse_vector(v1, 2, v2)
```

```
############################################################

def find_nonsingleton_words(text: str) -> Set[str]:
    """
    Cắt chuỗi text bởi khoảng trắng và trả về tập các từ mà xuất hiện hơn một lần.
    """
    # BEGIN_YOUR_CODE (our solution is 4 lines of code, but don't worry if you deviate from this)
```

```
    raise Exception("Not implemented yet")
    # END_YOUR_CODE
```

```
print(find_nonsingleton_words('con duong nay se dan den nha may duong'))
```

```
Start coding or generate with AI.
```