ชื่อ – นามสกุล วัฒชัย เตชะลือ รหัสนักศึกษา 65015137

Data Structures and Algorithm
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

การทดลองที่ 3 : Algorithm Analysis

จุดประสงค์

1. สามารถประเมิน Complexity ในโปรแกรมต่างๆ ได้

ตอนที่ 1 : Time Calculation

1. ให้นักศึกษาค้นหา Source Code ใน internet **ที่มีการทำงานเดียวกัน** 2 โปรแกรม แล้วใช้ฟังก์ชั่น time() เพื่อตรวจสอบกว่าการทำงานของทั้งสองโปรแกรม มีการใช้เวลาที่แตกต่างกันอย่างไรบ้าง

```
import time

st = time.time()
# source code
en = time.time()
elapsed_time = et – st
print('Execution time:', elapsed_time, 'seconds')
```

ตอบ

```python
import time

# โปรแกรมที่ 1
st = time.time()
for i in range(1000000):
    x = i**2
en = time.time()
elapsed_time1 = en - st
print('Execution time of program 1:', elapsed_time1, 'seconds')

# โปรแกรมที่ 2
st = time.time()
for i in range(1000000):
    x = i**3
en = time.time()
elapsed_time2 = en - st
print('Execution time of program 2:', elapsed_time2, 'seconds')

```

โปรแกรมทั้งสองชุดที่คุณให้มามีการทำงานที่แตกต่างกันเล็กน้อย โดยโปรแกรมที่หนึ่งมีการคำนวณ **x = i**2** ในแต่ละรอบของ loop ในขณะที่โปรแกรมที่สองมีการคำนวณ **x = i**3** ในแต่ละรอบของ loop. การคำนวณยกกำลังสามจะใช้เวลามากกว่าการคำนวณยกกำลังสอง ดังนั้นโปรแกรมที่สองอาจจะใช้เวลามากกว่าโปรแกรมที่หนึ่ง

ผลลัพธ์ที่ได้

```
pwsh kmitl_year2_work 128ms
& C:/Users/watthachai/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/watthachai/OneDrive/kmitl_year2_work/DSA/time.py
Execution time of program 1: 0.11768293380737305 seconds
Execution time of program 2: 0.1712353229522705 seconds
```

ชื่อ – นามสกุล วัฒชัย เตชะลือ รหัสนักศึกษา 65015137

2. ให้นักศึกษาทดสอบการทำงานของ ฟังก์ชั่น NearestNeighbors ของ sklearn.neighbors เมื่อมีการเรียกใช้งานเพื่อหา nearest point ของข้อมูลจุดทดสอบจำนวน 1000 จุด ในจุดอ้างอิงทั้งหมด 1,000,000 จุด ใน 2 วิธีการคือ การสร้างลูปเพื่อป้อนข้อมูลทดสอบทีละจุด จนครบ 1000 จุด กับการสร้าง list ของจุดทดสอบแล้วป้อนค่าดังกล่าวให้ library ทำงาน แล้วตอบคำถามว่าความเร็วในการทำงานของโปรแกรมส่วนที่เป็นตัวหนา เขียนโปรแกรมแบบไหนทำงานได้เร็วกว่า และเร็วกว่ากันกี่เท่า

โปรแกรมที่ 1	โปรแกรมที่ 2
import numpy as np from sklearn.neighbors import NearestNeighbors import random import time r = lambda: random.randint(0,10000) samples = [[r(),r(),r()] for i in range(1000000)] testdat = [[r(),r(),r()] for i in range(1000)] neigh = NearestNeighbors(n_neighbors=1) neigh.fit(samples) **detect1 = neigh.kneighbors(testdat)**	import numpy as np from sklearn.neighbors import NearestNeighbors import random import time r = lambda: random.randint(0,10000) samples = [[r(),r(),r()] for i in range(1000000)] testdat = [[r(),r(),r()] for i in range(1000)] neigh = NearestNeighbors(n_neighbors=1) neigh.fit(samples) **detect2=[]** **for i in testdat:** **detect2 += neigh.kneighbors([i])**

แล้วให้นักศึกษาลองวิเคราะห์ว่า การทำงานของทั้ง 2 โปรแกรมนั้นได้ผลลัพธ์ที่เหมือนกัน แต่เหตุใดจึงใช้เวลาในการทำงานที่แตกต่างกันมาก

ตอบ โค้ดทั้งสองชุดทำงานได้ผลลัพธ์เหมือนกัน แต่มีความแตกต่างในเวลาที่ใช้ในการทำงาน เนื่องจากวิธีการคำนวณของ `kneighbors` ในโค้ดที่สอง โดยใช้การวนซ้ำ (loop) และเรียกใช้ `kneighbors` แบบเดี่ยวๆ สำหรับแต่ละข้อมูลใน `testdat` ซึ่งจะใช้เวลามากกว่าการเรียกใช้ `kneighbors` แบบครั้งเดียวสำหรับข้อมูลทั้งหมดใน `testdat` ในโค้ดที่หนึ่ง การเรียกใช้ฟังก์ชันหลายครั้งจะมี overhead ที่เพิ่มขึ้นและส่งผลให้ใช้เวลามากขึ้นครับจาร

ชื่อ – นามสกุล วัฒชัย เตชะลือ รหัสนักศึกษา 65015137

ตอนที่ 2 : จงหา Big-O ของโปรแกรมต่อไปนี้

โปรแกรมที่ 1 :

```
n = input("input number : ")
n = int(n)
for i in range(1,n,2):
  print(i)
```

ตอบ O(n)

โปรแกรมที่ 2 :

```
n = input("input number : ")
n = int(n)
for x in range(1,n):
  for y in range(n - x):
    print (" ",end="")
  for y in range(1,x + 1):
    print(y,end="")
  for y in range(2,x + 1):
    print(x - y + 1,end="")
  print()
```

ตอบ O(n^2)

โปรแกรมที่ 3 :

```
n = input("input number : ")
n = int(n)
i=1
while i<n:
  print(i)
  i=2*i
```

ตอบ O(log n)

ชื่อ – นามสกุล วัฒชัย เตชะลือ รหัสนักศึกษา 65015137

ตอนที่ 3 : เขียนโปรแกรมค้นหาข้อมูลให้มี Big-O ตามที่กำหนด

1. กำหนดให้ตัวแปร data เป็นค่า list ของค่าสุ่ม

```
import random
rnddat = [random.randint(1,1000) for i in range(0,1000000)]
```

แล้วให้นักศึกษาเขียนโปรแกรมเรียงตัวเลขใน rnddat จากน้อยไปมากโดยไม่ใช้คำสั่ง sort() โดยให้การทำงานของโปรแกรมนี้มีค่า Big-O เป็น O(n^2)

ตอบ ผมใช้ Bubble Sort เพราะ มีค่า Big-O เป็น O(n^2) เพราะมีการวนซ้ำ (loop) สองชั้นซ้อนกัน และจำนวนครั้งที่ทำงานของ loop แต่ละชั้นเป็น n

```python
def bubble_sort(data):
    n = len(data)
    for i in range(n):
        for j in range(0, n-i-1):
            if data[j] > data[j+1]:
                data[j], data[j+1] = data[j+1], data[j]
    return data

# กำหนดให้ตัวแปร data เป็นค่า list ของค่าสุ่ม
import random
rnddat = [random.randint(1,1000) for i in range(0,1000000)]

# เรียงลำดับตัวเลขใน rnddat จากน้อยไปมาก
sorted_rnddat = bubble_sort(rnddat)

```

2. กำหนดให้ตัวแปร data มีค่าเป็น list ของตัวเลข 1-1,000,000

```
import random
dat = list(range(1,1000001))
```

แล้วให้นักศึกษาเขียนโปรแกรมรับ input เป็นตัวเลข 1 ตัวแล้วทำการค้นหาว่ามีตัวเลขดังกล่าวอยู่ที่ตำแหน่งใด โดยให้การทำงานของโปรแกรมนี้มีค่า Big-O เป็น O(log n)

ตอบ ผมใช้ Binary Search เพราะ มีค่า Big-O เป็น O(log n) เพราะใช้การวนซ้ำ (loop) while และใน loop จำนวนข้อมูลที่จะถูกค้นหาจะลดลงเป็นครึ่งในแต่ละรอบ จึงจำนวนครั้งที่ทำงานของ loop เป็น log n

ชื่อ – นามสกุล วัฒชัย เตชะลือ รหัสนักศึกษา 65015137

```python
def binary_search(data, target):
    low = 0
    high = len(data) - 1
    while low <= high:
        mid = (low + high) // 2
        if data[mid] == target:
            return mid
        elif data[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
    return -1

# กำหนดให้ตัวแปร data มีค่าเป็น list ของตัวเลข 1-1,000,000
import random
dat = list(range(1,1000001))

# รับ input เป็นตัวเลข 1 ตัว
target = int(input("Enter a number to search: "))

# ค้นหาตำแหน่งของตัวเลขใน dat
index = binary_search(dat, target)
if index != -1:
    print(f"The number {target} is found at index {index}")
else:
    print(f"The number {target} is not found in the list")

```