CPE231 Algorithm Design

Group เป็นคนไม่เอาถ่าน บ้านใช้เตาแก๊ส

Student ID 6507050 6026, 6507050 6037, 6507050 6039

Part I) From these simple codes, determine their asymptotic running time

1) nums is a list of size *n*

```
nums.append(1)
```

cost 1, No. of iteration 1, total = 1(1) = 1

O(1)

---

2) nums is a list of size *n*

```
nums.insert(0,2)
```

cost 1, No. of iteration 1, total = 1(1) = 1

O(1)

---

3) seq is a list with *n* elements

```
s = 0                    cost 1, No. of iteration 1
for x in seq:            cost 2, No. of iteration n+1
    s += x               cost 2, No. of iteration n
```

total = 1(1) + 2(n+1) + 2(n) = 4n + 3

O(n)

---

4) seq is a list with *n* elements

```
squares = [x**2 for x in seq]
```

```
for x in seq:            cost 2, No. of iteration n+1
    square = x**2        cost 2, No. of iteration n
```

total = 2(n+1) + 2n = 4n + 2

O(n)

---

5) seq is a list with *n* elements

```
s = 0                    cost 1, No. of iteration 1
for x in seq:            cost 2, No. of iteration n+1
    for y in seq:        cost 2, No. of iteration n(n+1)
        s += x*y         cost 3, No. of iteration n x n
```

total = 1(1) + 2(n+1) + n(n+1) + (n x n)

= 2n² + 3n + 1

O(n²)

---

6) seq is a list with *n* elements

```
s = 0                        cost 1, No. of iteration 1
for x in seq:                cost 2, No. of iteration n+1
    for y in seq:            cost 2, No. of iteration n(n+1)
        s += x*y             cost 3, No. of iteration n x n
    for z in seq:            cost 2, No. of iteration n(n+1)
        for w in seq:        cost 2, No. of iteration n(n)(n+1)
            s += x-w         cost 3, No. of iteration n x n x n
```

total = 1(1) + 2(n+1) + n(n+1) + (n x n) + n(n+1) + n(n)(n+1) + (n x n x n)

= 2n³ + 4n² + 4n + 1

O(n³)

---

7) seq1 contains n elements and seq2 contains m elements

```
s = 0                    cost 1, No. of iteration 1
for x in seq1:           cost 2, No. of iteration n+1
    for y in seq2:       cost 2, No. of iteration n(m+1)
        s+= x*y          cost 3, No. of iteration n x m
```

total = 1(1) + 2(n+1) + n(m+1) + (n x m)

= 2n² + 3n + 3

O(n²)

Group เป็นคนไม่เอาถ่าน บ้านใช้เตาแก๊ส
Student ID 6507050 6026, 6507050 6037, 6507050 6039

8)
```
s = 0                          cost   No. of Iteration
for seq2 in seq1:              1      1
    for x in seq2:             2      n+1
        s += x                 2      n(n+1)
                               2      nxn
```

total = 1(1) + 2(n+1) + 2(n)(n+1) + (nxn)
= $3n^2 + 4n + 3$

$O(n^2)$

---

9) seq is a list with *n* elements

```
s = 0                                  cost   No. of Iteration
n = len(seq)                   n       1      1
for i in range(n-1):           i, n    1      1
    for j in range(i+1, n):            2      n
        s += seq[i] * seq[j]           2      n(n-1)
                                       3      (n-1)(n-1)
```

total = 1(1) + 1(1) + 2n + 2(n)(n-1) + (n-1)(n-1)
= $3n^2 - 2n + 3$

$O(n^2)$

---

10) seq is a list with *n* elements

```
def sort_w_check(seq):
    n = len(seq)
    for i in range(n-1):
        if seq[i] > seq[i+1]:
            break
    else:
        return
    ...
```

Best case

cost	No. of Iteration
1	1
1	1
1	1

total = 1(1) + 1(1) + 1(1)
= 3

Worst case

cost	No. of Iteration
1	1
2	n
2	n
1	1

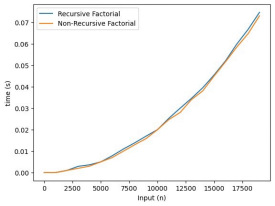
total = 1(1) + 2n + 2n + 1(1)
= 4n + 2

```
>>> sort_w_check(seq)
```

Best case : $O(1)$, Worst case : $O(n)$

Part II) Write code and do some experiments for these problem

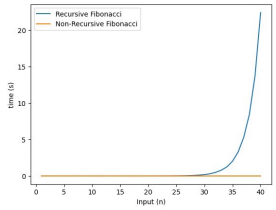
1) Run a timing experiment for the recursive and non-recursive algorithm for computing n! and discuss with the theorical estimation



จากกราฟจะเห็นว่าแบบ Recursive และ Non-recursive Factorial ใช้เวลาใกล้เคียงกัน ได้ time complexity คือ $O(n)$

---

2) Run a timing experiment for the recursive and non-recursive algorithm for computing Fibonacci(n) and discuss with the theorical estimation



จากกราฟจะเห็นว่าแบบ Recursive และ Non-recursive Fibonacci ใช้เวลาต่างกัน โดยแบบ Recursive จะใช้เวลาเพิ่มขึ้นเมื่อ $n > 30$ อย่างเห็นได้ชัด โดยได้ time complexity คือ $O(2^n)$ ในขณะที่แบบ Non-Recursive ใช้เวลาเข้าใกล้ 0 เสมอ จึงได้ time complexity คือ $O(1)$ ซึ่งดีกว่า

∴ การใช้ Algorithms ที่ต่างกันอาจทำให้ค่า Big O ออกมาต่างกัน จึงควรเลือก Algorithms ที่ได้ time complexity ดีที่สุด