

National University of Singapore
School of Computing
CS1010S: Programming Methodology
Semester I, 2017/2018

Mission 2 - Side Quest
Magic Efficiency

Release date: 01 September 2017

Due: 08 September 2017, 23:59

Required Files

- sidequest02.4-template.py

This side quest consists of **three** tasks.

Task 1: Simplification (4 marks)

Give the simplified big-O notations for all eight expressions below. Determine in each group which one has the faster-growing order of growth. (Note: you may express x^y in the format x^y)

- (i) $O(4^n n^2)$ vs $O(n 3^n)$
- (ii) $O(10000000000 n^2)$ vs $O(2^n / 10000000000)$
- (iii) $O(n^n + n^2 + 1)$ vs $O(4^n + 2^n)$
- (iv) $O(1^n)$ vs $O(n^2)$

Task 2: Analysis (2 marks)

Consider the following function foo:

```
def foo(n):  
    def bar(n):  
        if n == 0:  
            return 0  
        else:  
            return 1 + bar(n - 1)  
    return n * bar(n)
```

What is the time complexity for the running time of foo in terms of its input n ? What about space complexity?

Task 3: Improvisation (6 marks)

Consider the following two functions:

```
def bar(n):  
    if n == 0:  
        return 0  
    else:  
        return n + bar(n - 1)  
  
def foo(n):  
    if n == 0:  
        return 0  
    else:  
        return bar(n) + foo(n - 1)
```

- (i) What is the time complexity of bar? What about foo?
- (ii) What is the space complexity of bar? What about foo?
- (iii) Implement improved_foo **using any method** such that it computes the same value as foo, but with improved efficiency. To get full credit, your new function has to have improved (slower-growing) order of growth in both time **AND** space. Be sure that your function returns an Integer! Also, state the order of growths for your new function clearly in order notations.