CS1010S Tutorial 2

Sean Ng

AY2018/19 Sem 2, Week 4

Updated 2019-02-07 at 09:55:17

Materials: pengnam.github.io/CS1010S

Contact: `seanngpengnam@u.nus.edu`

Feedback From Coursemology

Complexity Analysis

```python
def repeat(i):
    if i<= 0:
        return '0'
    else:
        return str(i) + repeat(i-1)

# repeat(0) == '0'
# repeat(1) == '10'
# repeat(5) == '543210'
# repeat(8) == '9876543210'
```

What is the space and time complexity of this function?

(Assuming `str` is $O(1)$)

Complexity Analysis

In the return statement for `repeat(7)`,

1. `'7' + repeat(6)`

Complexity Analysis

In the return statement for `repeat(7)`,

1. `'7' + repeat(6)`
2. `'7' + '6' + repeat(5)`

Complexity Analysis

In the return statement for `repeat(7)`,

1. `'7' + repeat(6)`
2. `'7' + '6' + repeat(5)`
3. `'7' + '6' + '5' + repeat(4)`

Complexity Analysis

In the return statement for `repeat(7)`,

1. `’7’ + repeat(6)`
2. `’7’ + ’6’ + repeat(5)`
3. `’7’ + ’6’ + ’5’ + repeat(4)`
4. ...

Complexity Analysis

In the return statement for `repeat(7)`,

1. `’7’ + repeat(6)`
2. `’7’ + ’6’ + repeat(5)`
3. `’7’ + ’6’ + ’5’ + repeat(4)`
4. ...
5. `’7’ + ’6’ + ’5’ + ’4’ + ’3’ + ’2’ + ’1’ + ’0’`

The expression must be ‘expanded’ before the whole return statement is evaluated (‘completed’).

Complexity Analysis

```
def repeat2(i):
    result = ''
    for i in range(0, i+1):
        result = str(i) + result
    return result

# repeat(0) == '0'
# repeat(1) == '10'
# repeat(5) == '543210'
# repeat(8) == '9876543210'
```

What is the space and time complexity of this function?

(Assuming `str` is $O(1)$)

Complexity Analysis

In each iteration of the `for` loop in `repeat2(7)`,

1. `result = '0' + ''`

Complexity Analysis

In each iteration of the `for` loop in `repeat2(7)`,

1. `result = ’0’ + ’’`
2. `result = ’1’ + ’0’`

Complexity Analysis

In each iteration of the `for` loop in `repeat2(7)`,

1. `result = ’0’ + ’’`
2. `result = ’1’ + ’0’`
3. `result = ’2’ + ’10’`

Complexity Analysis

In each iteration of the `for` loop in `repeat2(7)`,

1. `result = '0' + ''`
2. `result = '1' + '0'`
3. `result = '2' + '10'`
4. `result = '3' + '210'`

Complexity Analysis

In each iteration of the `for` loop in `repeat2(7)`,

1. `result = ’0’ + ’’`
2. `result = ’1’ + ’0’`
3. `result = ’2’ + ’10’`
4. `result = ’3’ + ’210’`
5. `result = ’4’ + ’3210’`

Complexity Analysis

In each iteration of the `for` loop in `repeat2(7)`,

1. `result = ’0’ + ’’`
2. `result = ’1’ + ’0’`
3. `result = ’2’ + ’10’`
4. `result = ’3’ + ’210’`
5. `result = ’4’ + ’3210’`
6. ...
7. `result = ’7’ + ’6543210’`

The expression must be ‘expanded’ before the whole return statement is evaluated (‘completed’).

Complexity Analysis

```python
def super_repeat(i):
    if i<= 0:
        return '0'
    else:
        return repeat(i) + super_repeat(i-1)

# super_repeat(0) == '0'
# super_repeat(3) == '3210210100'
# super_repeat(5) == '543210432103210210100'
```

What is the space and time complexity of this function?

(Assuming `str` is $O(1)$)

Recursion & Iteration

- Closely related to *deferred operations*
- Recursive process:
  - The first term is first to be ‘written out’, but last to be evaluated
  - The last term is last to be ‘written out’, but first to be evaluated

1. `’7’ + repeat(6)`

Recursion & Iteration

- Closely related to *deferred operations*
- Recursive process:
    - The first term is first to be ‘written out’, but last to be evaluated
    - The last term is last to be ‘written out’, but first to be evaluated

1. `’7’ + repeat(6)`
2. `’7’ + ’6’ + repeat(5)`

Recursion & Iteration

- Closely related to *deferred operations*
- Recursive process:
  - The first term is first to be ‘written out’, but last to be evaluated
  - The last term is last to be ‘written out’, but first to be evaluated

1. `’7’ + repeat(6)`
2. `’7’ + ’6’ + repeat(5)`
3. `’7’ + ’6’ + ’5’ + repeat(4)`

Recursion & Iteration

- Closely related to *deferred operations*
- Recursive process:
    - The first term is first to be 'written out', but last to be evaluated
    - The last term is last to be 'written out', but first to be evaluated

1. `'7' + repeat(6)`
2. `'7' + '6' + repeat(5)`
3. `'7' + '6' + '5' + repeat(4)`
4. ...

Recursion & Iteration

- Closely related to *deferred operations*
- Recursive process:
    - The first term is first to be ‘written out’, but last to be evaluated
    - The last term is last to be ‘written out’, but first to be evaluated

1. `’7’ + repeat(6)`
2. `’7’ + ’6’ + repeat(5)`
3. `’7’ + ’6’ + ’5’ + repeat(4)`
4. ...
5. `’7’ + ’6’ + ’5’ + ’4’ + ’3’ + ’2’ + ’1’ + ’0’`

Recursion & Iteration

```
def my_sum(n):
    if n <= 0:
        return 0
    else:
        return n + my_sum(n-1)
```

Is this a recursive or iterative *process*?

Recursion & Iteration

```
def my_sum(n):
    result = 0
    for i in range(n+1):
        result = result + i
    return result
```

Is this a recursive or iterative *process*?

Recursion & Iteration

```
def my_sum(n):
    i = 0
    result = 0
    while i <= n:
        result = result + i
        i = i + 1
    return result
```

Is this a recursive or iterative *process*?

Recursion & Iteration

```
def my_sum(n):
    return helper(n, 0)

def helper(n, result):
    if n <= 0:
        return result
    else:
        return helper(n-1, result+n)
```

Is this a recursive or iterative *process*?

Tutorial

Question 1: Magnitude

Define a function `magnitude` that takes in the coordinates of two points on a plane: $(x1, y1)$ and $(x2, y2)$ as arguments and returns the magnitude of the vector between them.

Question 2: Area 1

A function can be viewed as a black box. All you need to know are the arguments it takes as input and what its output is.

One way of calculating the area of a triangle is using the formula

$$\text{area} = 1/2 \times \text{base} \times \text{height}$$

Define a function `area` that calculates and returns the area of any given triangle using this formula.

Decide what arguments it requires as input and what its return value should be.

Question 3: Area 2

Another way of calculating the area of a triangle with sides A, B, C is using the trigonometric ratio sine to get

$$\text{area} = 1/2 \times A \times B \times \sin(AB)$$

where AB is the included angle between sides A and B.

Define a function `area2` that calculates and returns the area of any given triangle using this formula.

Decide what arguments it requires as input and what its return value should be.

Both functions calculate the same result. Can they be directly substituted for each other? Why?

Question 4: Area 3

Assume you are given a function `herons_formula` that takes 3 arguments `a`, `b`, and `c` and returns the area of a triangle with sides of length `a`, `b`, and `c`

Define a function `area3` that uses Heron's formula to calculate and return the area of the given triangle given the x, y coordinates of the 3 points of the triangle.

You may use the `magnitude` function defined in Question 1.

Question 5: Expression Evaluation

2

```
def foo1():
    i = 0
    result = 0
    while i < 10:
        result += i
        i += 1
    return result
print(foo1())


def foo2():
    i = 0
    result = 0
    while i < 10:
        if i == 3:
            break
        result += i
        i += 1
    return result
print(foo2())
```

Question 6

```
def f(g):
    return g(2)
```

What happens if we ask the interpreter to evaluate the combination `f(f)`? Explain.

Question 7

Write a function `sum_even_factorials` that finds the sum of the factorials of the non-negative even numbers that are less than or equal to *n*.