ps 3 ... up for a few days

CSC165 fall 2019

counting steps...

Danny Heap
csc165-2019-09f@cs.toronto.edu
BA4270 (behind elevators)
Web page:
http://www.teach.cs.toronto.edu/~heap/165/F19/

Using Course notes: more Induction

nested loops

Make function variables "depend" on s_i steps

$\lceil \frac{n}{2} \rceil = (\frac{n}{2} + \varepsilon) n = \frac{n^2}{2} + n\varepsilon \rightarrow \Theta(n^2)$

```
def f5(n):
    for i in range(0, n, 2):
        for j in range(n):
            print(i - j)
```

1 step

$j = s_j$, $j \geq n$, $s_j \geq n$, $s = n$

$i = 2 s_i$, $i \geq n$, $2 s_i \geq n$, $s_i \geq n/2$

$\lceil n/2 \rceil$ = steps

n iterations × 1 step

n steps.

$\lceil n/2 \rceil$ × n steps

```
def f6(n):
    for i in range(n):
        for j in range(i):
            print(i - j)
```

1 step

$1 + 2 + \cdots + n-1 = \frac{n(n-1)}{2} = \frac{n^2 - n}{2} = \frac{n^2}{2} - \frac{n}{2}$

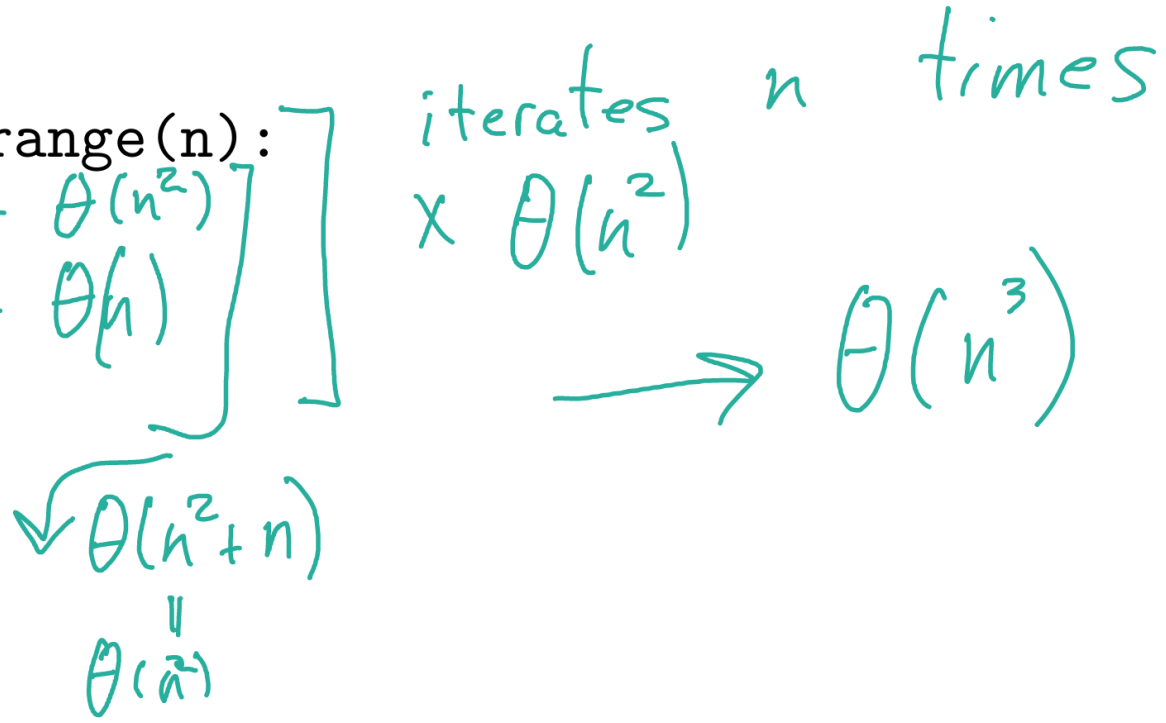
$= \frac{n^2}{4} + \left(\frac{n^2}{4} - \frac{n}{2}\right) \geq \frac{1}{4} n^2$, for $n \geq 2$.

$O(n^2)$ .... $c = 1$

$\Omega(n^2)$ $c = 1/4$, $n_0 = 2$

composition, combination

```
def f7(n):
    for k in range(n):
        f6(n)
        f2(n)
```



```
def twisty3(n):
    steps = 0
    while n > 0:
        if n % 3 == 0:
            n = n // 3
        elif n % 3 == 1:
            n = 3*n - 3
        else:
            n = 3*n - 6
        steps = steps + 1
        print(n, steps)
    return steps
```

n	steps
0	0
1	1
2	1
3	2
4	4
5	4
6	2
7	4?

clumsy is_prime

better ways to test for primality

7 steps - popular demand!

```
def is_prime(n):
    if n < 2:
        return False
    else:
        for d in range(2,n):
            if n % d == 0:
                return False
        return True
```

n	$RT_{ip}(n)$
5	3+7 - 10 steps
⋮	
35	4+7 - 11 steps
36	1+7 - 8 steps
37	35+7 - 42
	O(n)
	Ω(1)