CSC165 fall 2019

counting steps...

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

Using Course notes: algorithm analysis

Computer Science
UNIVERSITY OF TORONTO

Outline

notes

how to choose n so that $2^n \geq cn^{1000}$

a couple of hints...

big-Theta means...

$$g \in \Theta(f) : \exists c_1, c_2, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0$$
$$\Rightarrow g(n) \leq c_1 f(n) \wedge g(n) \geq c_2 f(n)$$

products and sums — redux

- $f + g$
- af
- $f \cdot g$

back to code...

“steps” and input “size”

counting loops...

```
def f0(n):
    x = n
    print(x * 2)
    return x + 3
```

```
def f1(n):
    for i in range(10):
        print(n)
```

```
def f2(n):
    for i in range(n):
        print(n)


def f3(n):
    i = 0
    while i*i < n:
        print(i)
        i = i + 1


def f4(n):
    i = 0
    while i**(1/2) < n:
        print(2*i)
        i = i + 1
```

nested loops

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

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

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
```

clumsy is_prime

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

Notes