CSC165 fall 2019

end induction...

...begin algorithm analysis

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: ~~more~~ Induction & algorithm Analysis

every set with n elements has 2^n subsets

order of introductions... and intuition

Contrast to Ex 3.4

$$P(n): \forall \text{ set } s,\ |S| = n \Rightarrow |\mathcal{P}(s)| = 2^n$$

$$A = \{a, b, c\}$$

$$\mathcal{P}(\emptyset) = \{\emptyset\}$$

$$\mathcal{P}(\{a\}) = \{\emptyset, \{a\}\}$$

$$\mathcal{P}(\{a,b\}) = \{\emptyset, \{a\}, \{b\}, \{a,b\}\}$$

$$\mathcal{P}(\{a,b,c\}) = \{\emptyset, \{a\}, \{b\}, \{a,b\}, \{c\}, \{a,c\}, \{b,c\}, \{a,b,c\}\}$$

$$|A| = k+1$$

taking binary representation apart

$(11)_2$ $(11\bullet1)_2$

5378×10

$\to 0$

suppose n is a natural number with binary representation:

$$n = \Sigma_{i=0}^{i=p} b_i 2^i = \boxed{b_p 2^p + b_{p-1} 2^{p-1} + \cdots + 2^1 b_1 + 2^0 b_0}$$

// 2

figure out b_0 based on whether n is odd or even... suggesting

$b_p 2^{p-1} + b_{p-1} 2^{p-2} + \cdots + b_1 2^0 = \lfloor \frac{n}{2} \rfloor$

```
def natural_to_binary(n: int) -> str:
    # convert n to equivalent binary string
    bs = str(n % 2)
    n = n // 2
    while n > 0:
        bs = str(n % 2) + bs
        n = n // 2
    return bs
```

figure out right-most digit

$n \gg 1$

$n \ll 1$

throw away right-most digit,

two's complement

time resource

really care about use of time resource.

measure by wall clock.

How much time does this take?

how quickly does run-time grow with input?

```
def f(list_):
    for i in list_:
        print(i)
```

gnarly details

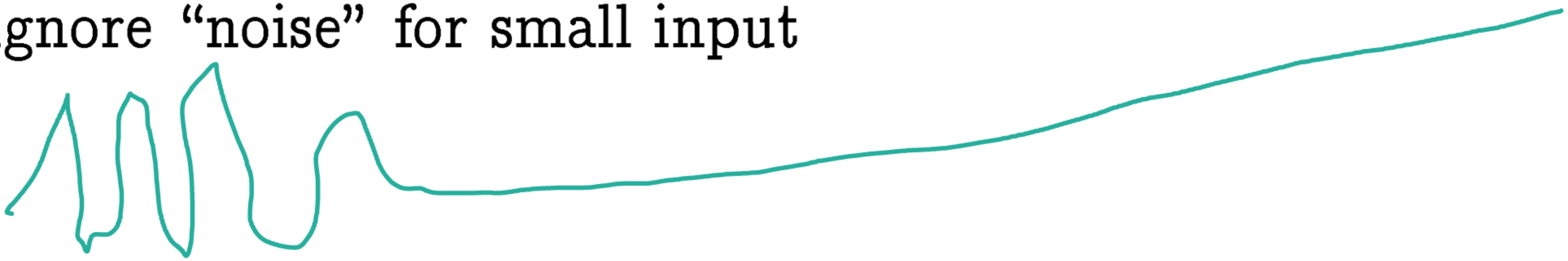
does $x < 3$ execute faster/slower than $x = 3$.

don't care if don't depend on input size

assumptions, assumptions...

Want run-time as a function of n
e.g. $t(n)$

- "steps" operation that don't depend on input size (often n).
- ignore constant factors
  multiply some $c \in \mathbb{R}^+$
- ignore "noise" for small input



We care about growth rate of time consumption

formalizing assumptions

$g(n)$ is our run time function

- f absolutely dominates g

$\forall n \in \mathbb{N}, g(n) \leq f(n).$

- f dominates g up to a constant factor

$\exists c \in \mathbb{R}^+, \forall n \in \mathbb{N}, g(n) \leq c f(n)$

- f eventually dominates g up to a constant factor

$\exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0 \Longrightarrow g(n) \leq c f(n)$

What should domain and range of f, g be?

$f, g : \mathbb{N} \to \mathbb{R}^{\geq 0}$

big-Oh, big-Omega, big-Theta

... and you're started on the Greek alphabet...

f is $\Theta(g)$

$$f \in \mathcal{O}(g) : \exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow f(n) \leq cg(n)$$

try to show that $\forall a, b \in \mathbb{R}^+, an + b \in \mathcal{O}(n^2)$

$\forall a, b \in \mathbb{R}^+, \exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow an + b \leq cn^2$

Let $a, b \in \mathbb{R}^+$. Let $c =$ ________.
Let $n_0 =$ ________. Let $n \in \mathbb{N}$. assume $n \geq n_0$. WTS $an + b \leq cn^2$.

→ you do body(s)

try choosing n_0 big enough.

rough work.

$c'n^2 \geq an$ | $c' = a$

$c''n^2 \geq b$ | $c'' = b$

$c = c' + c'' = a + b$

Computer Science
UNIVERSITY OF TORONTO