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
every set with \( n \) elements has \( 2^n \) subsets.

Order of introductions... and intuition

\[ P(n) : \forall \text{set } s, |s| = n \implies |P(s)| = 2^n \]

\[
\begin{align*}
A &= \{a, b, c\} \\
\mathcal{P}(\emptyset) &= \{\emptyset\} \\
\mathcal{P}(\{a\}) &= \{\emptyset, \{a\}\} \\
\mathcal{P}(\{a, b\}) &= \{\emptyset, \{a\}, \{b\}, \{a, b\}\} \\
\mathcal{P}(\{a, b, c\}) &= \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}
\end{align*}
\]
taking binary representation apart

\[(11)_{2} \quad (1101)_{2}\]

suppose \( n \) is a natural number with binary representation:

\[n = \sum_{i=0}^{i=p} b_i 2^i = b_p 2^p + b_{p-1}2^{p-1} + \cdots + 2^1 b_1 + 2^0 b_0\]

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

\[b_p 2^{p-1} + b_{p-1}2^{p-2} + \cdots + b_1 2^0 = \left\lfloor \frac{n}{2} \right\rfloor\]

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
time resourceeally care about use
of time resource.

How much time does this take?

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

gnarly details

measure by wall clock.

how quickly does run-time grow with input?

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

- $f$ absolutely dominates $g$
  \[
  \forall n \in \mathbb{N}, \ g(n) \leq f(n).
  \]

- $f$ dominates $g$ up to a constant factor
  \[
  \exists c \in \mathbb{R}^+, \ \forall n \in \mathbb{N}, \ g(n) \leq c \cdot f(n).
  \]

- $f$ eventually dominates $g$ up to a constant factor
  \[
  \exists c, n_0 \in \mathbb{R}^+, \ \forall n \in \mathbb{N}, \ n \geq n_0 \implies g(n) \leq c \cdot f(n).
  \]

What should domain and range of $f, g$ be?

$f, g : \mathbb{N} \rightarrow \mathbb{R}^+$
big-Oh, big-Omega, big-Theta
... and you’re started on the Greek alphabet...

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

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

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

Let \( a, b \in \mathbb{R}^+ \). Let \( c = \frac{b}{a} \).

Let \( n_0 = \_ \). Let \( n \in \mathbb{N} \). Assume \( n \geq n_0 \). WTS \( an + b \leq cn^2 \).

\[ \text{you do body(s)} \]

\[ \text{try choosing} \quad \text{no big enough} \]

rough work:
\[ \begin{align*}
  cn^2 & \geq an \quad | c = a \\
  cn^2 & \geq b \quad | c = b \\
  c & = c' + c'' = a + b
\end{align*} \]