CSCI 3104
Lecture 7: Hash Tables Continued

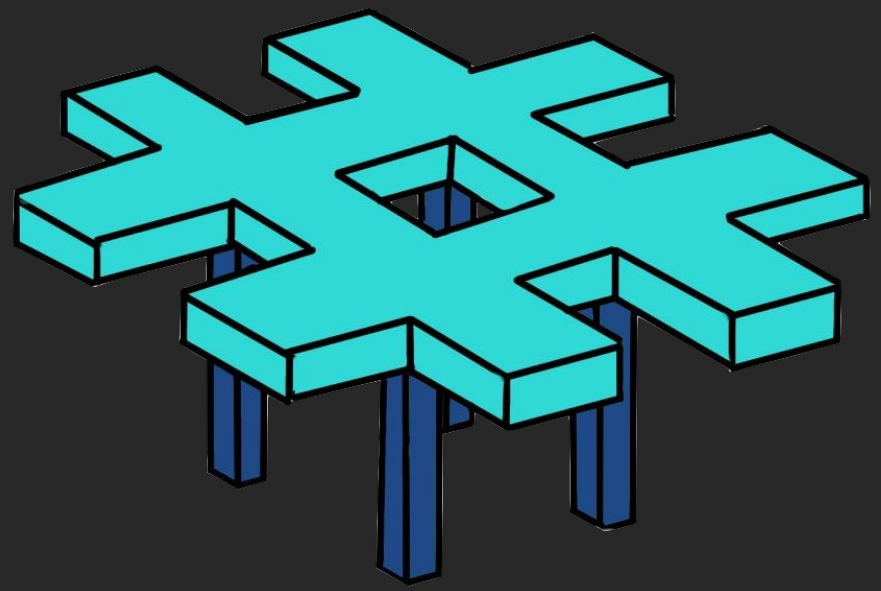
Caleb Escobedo

Caleb.Escobedo@colorado.edu

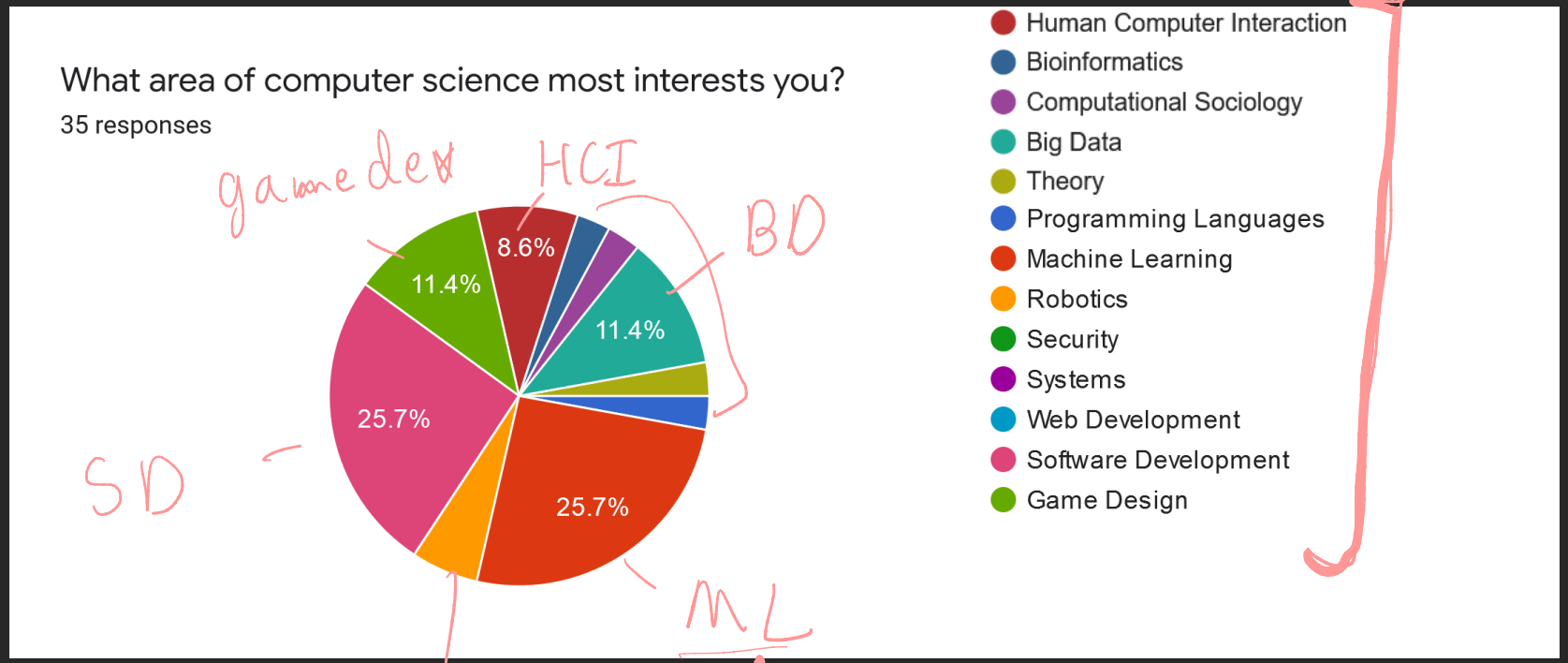
Lecture Outline

- Poll review/Administrative
- Hash Table Collisions
  - Problem
  - Solution: Dynamic Resizing
  - Amortized Analysis
- Hash Functions – Quick
- 3 Example Problems for HW2A

T(n)



Poll Review!


What area of computer science most interests you?
35 responses
8.6%
11.4%
11.4%
25.7%
25.7%
Human Computer Interaction
Bioinformatics
Computational Sociology
Big Data
Theory
Programming Languages
Machine Learning
Robotics
Security
Systems
Web Development
Software Development
Game Design
gamedev
HCI
BD
SD
ML
Robotics
NLP
CV
RL

Administrative

- Positive response to examples: I'll keep them coming!
- Homework 1B due tonight at 11:55!
- Homework 1A grades out my tomorrow
- Homework 2B released today Quick sort
- Everyone did GREAT on the Leetcode problem for HW1A

Administrative

1) $n^{\log_b(a+\varepsilon)}$

↳ $n^{\log_b(a)+\varepsilon}$

- Correction from last class: epsilon is not in the in the log

Master theorem. Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function and let $T(n)$ be defined on the non-negative integers by the recurrence,

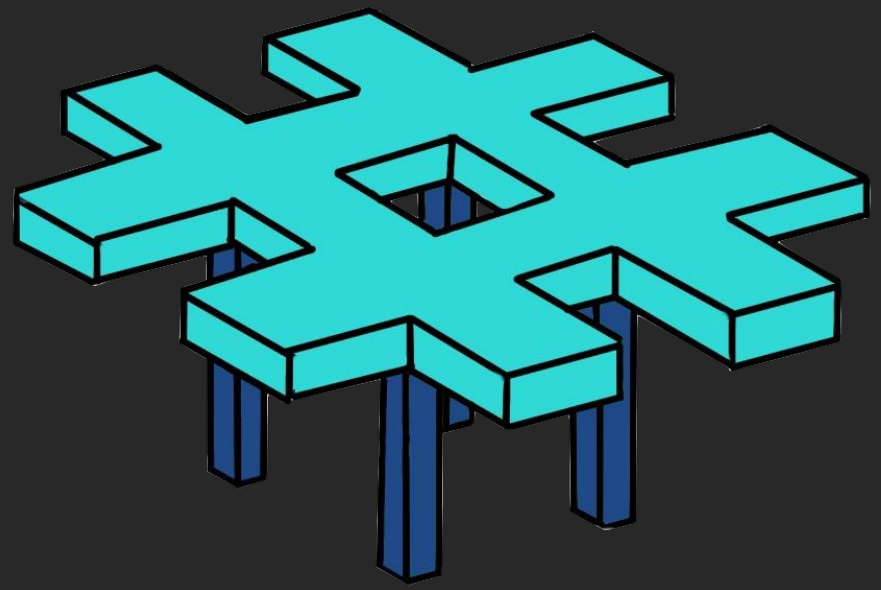
$$T(n) = a\,T(n/b) + f(n) \ ,$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ can be bounded asymptotically as follows:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $a\,f(n/b) \leq c\,f(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. □

Lecture Outline

- ~~Poll review/Administrative~~
- Hash Table Collisions
  - Problem
  - Solution: Dynamic Resizing
  - Amortized Analysis
- Hash Functions
- 3 Example Problems for HW2A



Hash Collision Review

- How does this function grow in as *n* increases? n^2
- How does this function decrease as *ℓ increases?* $O(n)$

$$
\begin{aligned}
E(X) &= E\left(\sum_{ij} X_{ij}\right) \\
&= \sum_{ij} E(X_{ij}) \\
&= \sum_{ij} \frac{1}{\ell} \\
&= \binom{n}{2} \cdot \frac{1}{\ell} \\
&= \frac{n(n-1)}{2\ell} \cdot
\end{aligned}
$$

ℓ = table size
n = # items

$\ell = 365$

Ex) n = 55 41.1

Ex) n = 200 54.5

Problems We Run Into

α = load factor

- If the size of ℓ is pre-allocated, we run into a problem as we take the limit of $n \to \infty$

$O(1+\alpha)$ $O(1+C(n))$

$\frac{n}{\ell} = \alpha$ — n: # of items; ℓ: size of table

- The runtime of every operation grows linearly with n!

$O(n)$

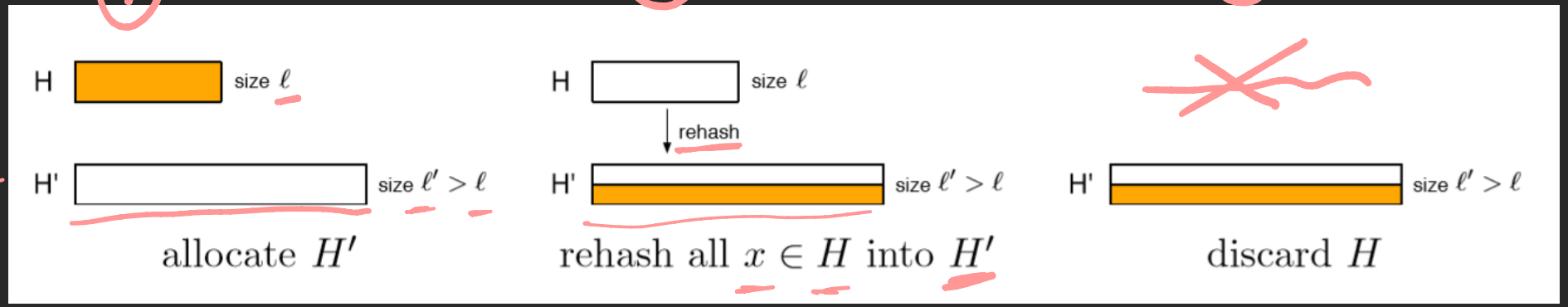
- If we knew how many add operations would be done we could allocation $\ell >> n$ as the size of our table. Is that feasible??

NO

$\frac{n}{\ell} = \frac{1}{\ell} \cdot n = C \cdot n$

Solution: *Dynamic Resizing!*

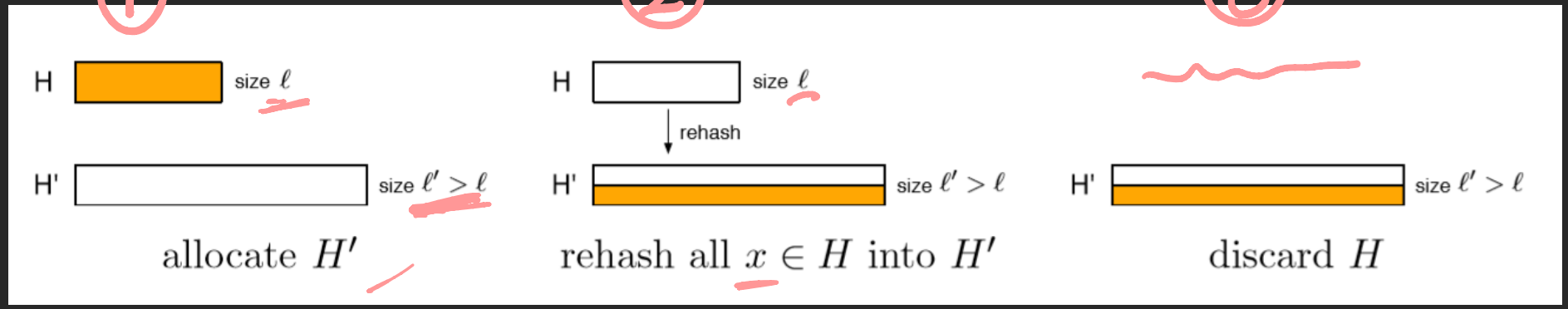
- Same example from the start of last class, but with a hash table



$\forall x \; h''(x) \to idx$

Cost Analysis: Dynamic Resizing!

- We compare the cost of some very large operations (resize) and other very small operations (add) using *amortized analysis:*
  - Goal: Show that by occasionally paying a large cost, we can ensure that other costs remain very small



Solution: *Dynamic Resizing!*

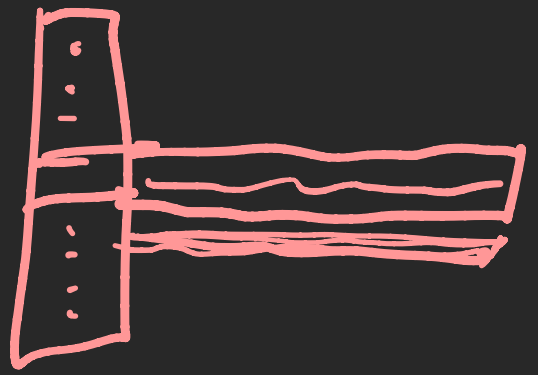
- What is the total cost T(n) of some sequence of operations?
  - the sum costs of all resizing events
  - the sum cost of all individual operations
- Note that - the amortized cost of each individual operation is:

$$T(n)/n$$

total run time

# of operations

Double or Nothing



- We pick a maximum load factor for our Hash table:

$$\alpha_{\max} = O(1) = c'$$

$$\frac{n}{\ell} = \alpha$$

- Whenever we reach this max value, we allocate a new hash table

$$\ell' = 2\ell$$

Analysis

$\ell = 1$

- We pick a maximum load factor for our Hash table:

$$\alpha_{\max} = O(1) = c'$$

$$c' = 1$$

$$1 + 2 + 4 + 8 + \cdots \frac{n}{2} + n = \sum_{i=1}^{k} 2^i$$

Analysis

- The cost off all resizing events is:

$$1 + 2 + 4 + 8 + \cdots + \frac{n}{2} + n = \sum_{i=1}^{k} 2^i$$

- What is k? How often do we need to increase the size of our array:

$$\sum_{i=1}^{\log_2 n} 2^i \leq 2n$$

Analysis

- The total cost of all operations + all resizing events is:

O(1)

all ops

$$T(n) = n + 2n = \Theta(n)$$

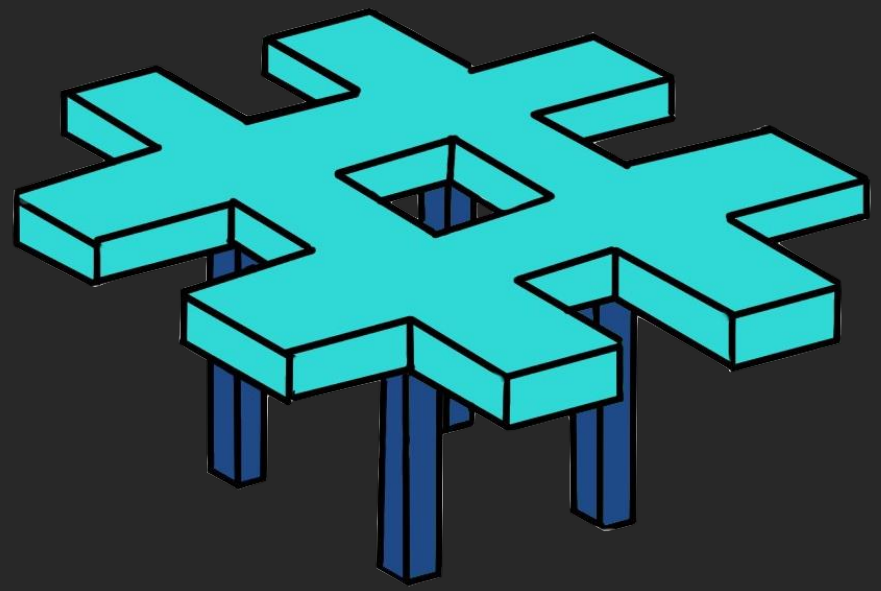
- What is the cost per operation?

$T(n)/n$

$$\frac{n+2n}{n} = \frac{1+2}{1} = 3 = \Theta(1)$$

Lecture Outline

- ~~Poll review/Administrative~~
- ~~Hash Table Collisions~~
  - ~~Problem~~
  - ~~Solution: Dynamic Resizing~~
  - ~~Amortized Analysis~~
- Hash Functions
- 3 Example Problems for HW2A



Hash Functions

$$\mathbb{N} = \{0, 1, 2, \ldots\}$$

- Division Method:

key → k ; size of table → m

$$h(k) = k \bmod m$$

- Ch 11.3.1 page 263

$m \neq 2^p$

Hash Functions

$$\mathbb{N} = \{0, 1, 2, \ldots\}$$

- Multiplication Method:

$$h(k) = \lfloor m\,(kA \bmod 1) \rfloor$$

size of table key $0 < A < 1$

$kA = 4{,}23\ldots$

- Ch 11.3.2 page 264

Hash Functions

$$\mathbb{N} = \{0, 1, 2, \ldots\}$$

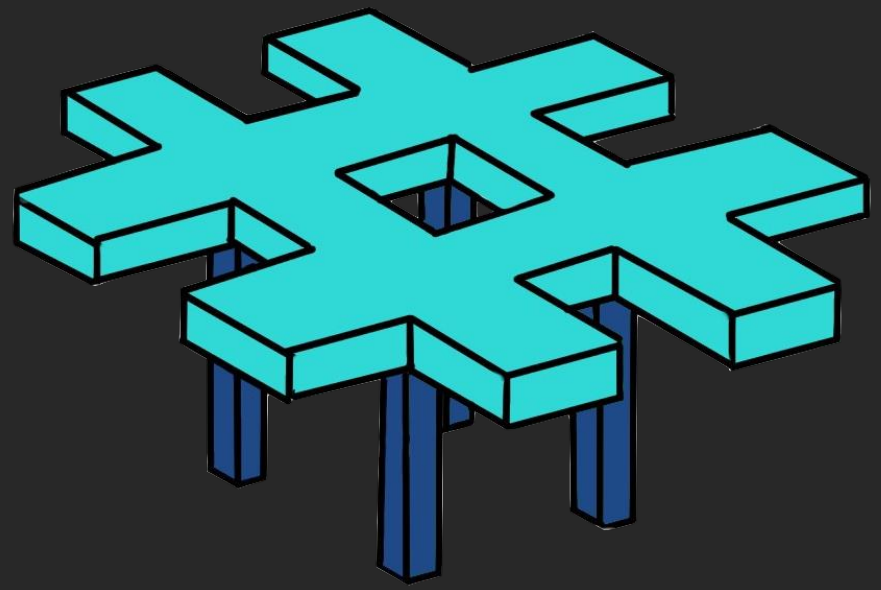
- Universal Hashing: using the other methods an adversary can still cause our runtime to be Θ(n) by using specific key values
- Use Randomness to help prevent this case by choosing a different hash function independent of the key

$$h \in \mathcal{H}$$

- Ch. 11.3.3 page 265

Lecture Outline

- ~~Poll review/Administrative~~
- ~~Hash Table Collisions~~
  - ~~Problem~~
  - ~~Solution: Dynamic Resizing~~
  - ~~Amortized Analysis~~
- ~~Hash Functions~~
- 3 Example Problems for HW2A



Can we use →

Example 1

why not?

- $T(n) = 9T(n/3) + n^2\log(n)$

Master theorem. Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function and let $T(n)$ be defined on the non-negative integers by the recurrence,

$$T(n) = a\,T(n/b) + f(n)\ ,$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ can be bounded asymptotically as follows:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $a\,f(n/b) \leq c\,f(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. □

$f(n) = \Omega(g(n))$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)}$$

$$\lim_{n\to\infty} \frac{n^2\log n}{n^{2+\epsilon}}$$

$$\lim_{n\to\infty} \frac{n^2\log n}{n^2 \cdot n^{\epsilon}}$$

$$\lim_{n\to\infty} \frac{\log n}{n^{\epsilon}} \xrightarrow{\frac{d}{dn}} \lim_{n\to\infty} \frac{\frac{1}{n}}{\epsilon n^{\epsilon-1}}$$

$$\lim_{n\to\infty} \frac{1}{\epsilon n^{\epsilon}} = 0$$

$$\frac{1}{n^1 \cdot \epsilon n^{\epsilon-1}}$$

1) $n^{\log_b a} = n^{\log_3 9} = n^2$

2) $f(n) = n^2 \log n$

3) compare $f(n)$ and $n^{\log_b a}$

$$n^2\log n = n^{2+\epsilon}$$

$$\frac{d}{dn}\log n = \frac{1}{n}$$

Example 2

a = How many recursive calls?
b) how we divide our input

- What is the runtime of this code in terms of n?

```
def fun(n):
  if(n >=0):
    print("Hello CSCI3104!")
  1 fun(n/4)
  2 fun(n/4)
  3 fun(n/4)
```

Master theorem. Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function and let $T(n)$ be defined on the non-negative integers by the recurrence,

$$T(n) = a\,T(n/b) + f(n)\ ,$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ can be bounded asymptotically as follows:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $a\,f(n/b) \leq c\,f(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. □

$T(n) = 3T(n/4) + \Theta(1)$

$T(n) = \Theta(n^{\log_4 3})$

$\varepsilon = \log_4 3$

1) $n^{\log_4 3}$
2) $\Theta(1)$
$1 = n^{(\log_4 3) - \varepsilon}$

Example 3

- What is the runtime of this code in terms of n?

$$\varepsilon = \frac{\log\left(\frac{4}{3}\right)}{\log(4)}$$

```python
def alsofun(n):
  if(n >=0):
    for i in range(n/2):
      print("Hello CSCI3104!")
    alsofun(n/4)
    alsofun(n/4)
    alsofun(n/4)
```

Master theorem. Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function and let $T(n)$ be defined on the non-negative integers by the recurrence,

$$T(n) = a\,T(n/b) + f(n) \ ,$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ can be bounded asymptotically as follows:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $a\,f(n/b) \leq c\,f(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. □

$3\frac{n}{4} \leq cn$

$\frac{3}{4} \leq c < 1$ $\quad c = \frac{3}{4}$

$\log_4 3 + \varepsilon = 1$

$\varepsilon = 1 - \log_4 3$

$T(n) = \Theta(n)$