Graduate Algorithms.
Georgia Institute of Technology.

Lecture 0

Logistic and organization.

- **Instructor**: Gerandy Brito (you can call me Gerandy, or Brito, or Prof. Brito.)
- **TAs**: TBA on canvas.
- Webpage for the class: **CANVAS**. Everything will be posted there (including these slides!)

Logistic and organization.

- **Lectures time and location**: T TH: 9:30am-10:45am on Bluejeans and sometimes pre-recorded.
- **Syllabus**: on canvas. You are responsible for reading it.
- **Schedule**: A file with a schedule is on canvas and will be frequently updated.

Communications.

- **Piazza**: is great for discussion with your peers. TAs will be following the threads.
- Due to the large size of my classes, I encourage you to use piazza to contact me (email takes me longer to respond!).

Evaluations.

HOMEWORKS.

- Weekly-ish homeworks. 40% of final grade.
- Each homework will be **posted on gradescope**. You wil type your answers in.
- **Due date**: HW will be posted on a Thursday, due the following Thursday before lecture.
- **Late submission** the Sunday after due day (10% penalty). Only under very special circumstances a submission will be accepted after that.

Evaluations.

Participation.

- 40% of your final grade.
- During lectures, a multiple choice question. Answers must be submitted on canvas.
- You need to attend/watch the lectures to answer it.

Evaluations.

Midterm.

- 20% of your final grade.
- During lecture, see the schedule for date and content to be covered.
- Exam will be proctored online, using Honorlock (on canvas).

Before the final exam.

Weighting your assingments.

Homeworks: 40%.
Participation: 40%.
Midterm: 20%.
Letter grades: A $[90, 100]$, B $[80, 89.9]$, C $[70, 79.9]$, D $[60, 69.9]$, F $[0, 59.9]$.

The Final redemption

- Final Exam will be cumulative.
- See the official calendar for date and time.
- Final Exam will substitute your midterm (if higher).

Topics we will cover

- (Advanced) Divide and conquer: median of medians, Fast Fourier Transform.
- Arithmetic and algorithms, RSA cryptosystems.
- Graphs algorithms: spanning trees, matchings, Markov Chains, flow networks.

Divide and Conquer.

Big-O notation.

Definition

A function $f(n)$ is said to be *big O* of the function $g(n)$ (denoted as $f(n) = O(g(n))$ or $f(n) \preceq g(n)$) if there exists constants C and N such that:

$$|f(n)| \leq C|g(n)| \quad \text{for all } n \geq N.$$

Big-O notation: examples

- $f(n) = n, \; g(n) = n^2$.
- $f(n) = n + 3, \; g(n) = n^2 - 1$.
- $f(n) = n + 1, \; g(n) = 2n + 5$.
- $f(n) = \log(n), \; g(n) = n^{0.5}$.
- $f(n) = n^4, \; g(n) = 2^n$.

Hierarchy of fundamental functions

For $\alpha \leq 1 \leq \beta < \gamma$ and $a > 1$

$$\log(n) \preceq n^{\alpha} \preceq n \preceq n\log(n) \preceq n^{\beta}\log(n) \preceq n^{\gamma} \preceq a^{n} \preceq n!.$$

We want to be optimal, in the worth case possible.

Example

Given a number a and a natural number $n > 0$, design an algorithm to compute a^n. Assume your machine does multiplication in $O(1)$.

Simpler (naive) approach:

Power(a,n)

```
c=a
For i = 1 to n − 1:
   c = c ∗ a:
return c
```

$T(n) = O(n)$.

We want to be optimal, in the worth case possible.

Example

Given a number a and a natural number $n > 0$, design an algorithm to compute a^n. Assume your machine does multiplication in $O(1)$.

Observation: for n even: $a^n = a^{n/2} * a^{n/2}$.

Power(a, n)
If $n = 1$ return a.
If n is even:
  $x =$ **Power**$(a, n/2)$
return $x * x$
else:
  $x =$ **Power**$(a, (n-1)/2)$
return $a * x * x$

$T(n) = ?$

We want to be optimal, in the worth case possible.

Example

Given a number a and a natural number $n > 0$, design an algorithm to compute a^n. Assume your machine does multiplication in $O(1)$.

Power(a, n)
`If` $n = 1$ `return` a.
`If` n is even:
  $x =$ **Power**$(a, n/2)$
`return` $x * x$
`else`:
  $x =$ **Power**$(a, (n-1)/2)$
`return` $a * x * x$
$T(n) = ?$
Hint: To compute a^8 we call the algorithm on $n = 4$, $n = 2$, $n = 1$.

Example

Given a number a and a natural number $n > 0$, design an algorithm to compute a^n. Assume your machine does multiplication in $O(1)$.

Power(a, n)
If $n = 1$ `return` a.
If n is even:
  $x =$ **Power**$(a, n/2)$
`return` $x * x$
`else`:
  $x =$ **Power**$(a, (n-1)/2)$
`return` $a * x * x$

$T(n) = T(n/2) + O(1) \to T(n) = O(\log(n))$.

Master Theorem

Theorem

Let a, b, d be constants, not necessarily integers, with $a > 0,\ b > 1$ and $d \geq 0$. If $T(n) = aT(n/b) + O(n^d)$ then

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b(a); \\ O(n^d \log(n)), & \text{if } d = \log_b(a); \\ O(n^{\log_b(a)}), & \text{if } d < \log_b(a). \end{cases}$$

Before: $T(n) = T(n/2) + O(1)$.
$a = 1,\ b = 2,\ d = 0$ ($0 = d = \log_b(a) = \log_2(1)$)
$T(n) = O(n^0 \log(n))$ by the Master Theorem.

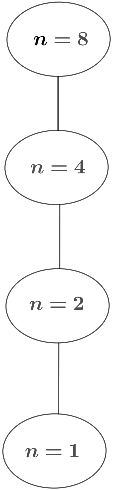
Poll 1

Question

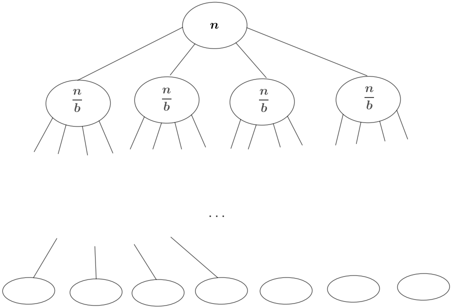
The solution to the recurrence $T(n) = 3T(n/2) + O(n)$ is

- A $O(n)$.
- B $O(n^{\log_2(3)})$.
- C $O(n \log(n))$.
- D $O(n^2)$.

Understanding the running time of D&C



Understanding the running time of D&C



$\leftarrow$ 1st level

$\leftarrow$ kth level

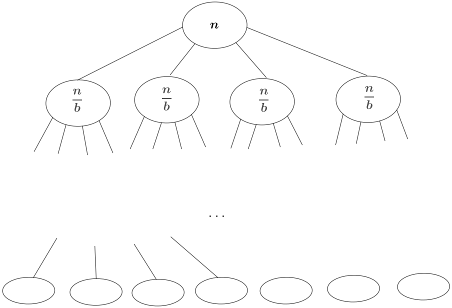
$Number\ of\ subproblems: \ a^k$

$Input\ size: \ \frac{n}{b^k}$

$Number\ of\ operations: \ a^{k-1} \times O\left(\frac{n}{b^k}\right)^d$

$\leftarrow$ $\log_b(n)^{th}$ level *base case!*

Understanding the running time of D&C



$\leftarrow$ 1st level

$\leftarrow$ kth level

$Number\ of\ subproblems:\ a^k$

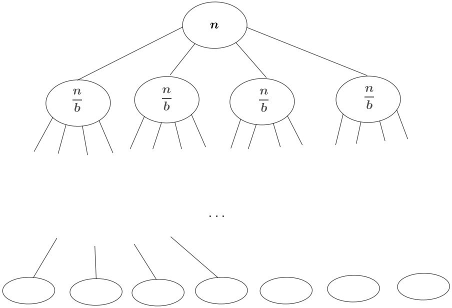
$Input\ size:\ \frac{n}{b^k}$

$Number\ of\ operations:\ a^{k-1} \times O\left(\frac{n}{b^k}\right)^d$

$\leftarrow$ $\log_b(n)$th level $\quad$ *base case!*

$$a^{\log_b(n)-1} O\left(\frac{n}{b^{\log_b(n)}}\right)^d + a^{\log_b(n)-2} O\left(\frac{n}{b^{\log_b(n)-1}}\right)^d + \cdots + O\left(\frac{n}{b}\right)^d.$$

Understanding the running time of D&C



$\leftarrow$ 1^{st} level

$\leftarrow$ k^{th} level

$Number\ of\ subproblems:\ a^k$

$Input\ size:\ \frac{n}{b^k}$

$Number\ of\ operations:\ a^{k-1} \times O\left(\frac{n}{b^k}\right)^d$

$\leftarrow$ $\log_b(n)^{th}$ level $\quad$ *base case!*

$$\sum_{k=1}^{\log_b(n)} a^{k-1} O\left(\frac{n}{b^k}\right)^d = C \sum_{k=1}^{\log_b(n)} n^d \left(\frac{a}{b^d}\right)^k = Cn^d \frac{r^{\log_b(n)+1} - 1}{r - 1}$$

Master Theorem

Theorem

Let a, b, d be constants, not necessarily integers, with $a > 0$, $b > 1$ and $d \geq 0$. If $T(n) = aT(n/b) + O(n^d)$ then

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b(a); \\ O(n^d \log(n)), & \text{if } d = \log_b(a); \\ O(n^{\log_b(a)}), & \text{if } d < \log_b(a). \end{cases}$$

$$T(n) = O(n^d r^{\log_b(n)}) \text{ for } r = \frac{a}{b^d}$$

Note:

$$r < 1 \text{ if and only if } a < b^d \text{ if and only if } \log_b(a) < d.$$
$$r = 1 \text{ if and only if } a = b^d \text{ if and only if } \log_b(a) = d.$$
$$r > 1 \text{ if and only if } a > b^d \text{ if and only if } \log_b(a) > d.$$