

Lecture 31

CSE 331

Nov 14, 2022

Homework 6 reminder

Homework 6

- Part **(b)**: Present a divide and conquer algorithm that given non-negative integers a and n computes `Power` (a, n) in $O(\log n)$ time.

Important Note

To get credit you must present a recursive divide and conquer algorithm and then analyze its running time by solving a recurrence relation. If you present an algorithm that is not a divide and conquer algorithm you will get a level 0 on this entire part.

Question 1 (Exponentiation) [50 points]

The Problem

We will consider the problem of exponentiating an integer to another. In particular, for non-negative integers a and n , define `Power` (a, n) be the number a^n . (For this problem assume that you can multiply two integers in $O(1)$ time.) Here are the two parts of the problem:

- Part **(a)**: Present a naive algorithm that given non-negative integers a and n computes `Power` (a, n) in time $O(n)$.

Note

For this part, there is no need to prove correctness of the naive algorithm but you do need a runtime analysis.

- Part **(b)**: Present a divide and conquer algorithm that given non-negative integers a and n computes `Power` (a, n) in $O(\log n)$ time.

A dope panel TOMORROW!



OUR AWESOME PHD STUDENT SPEAKERS

How do I know if I want to do a PhD, and how do I get there?



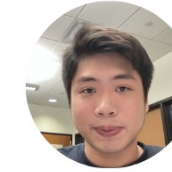
JESS GROGAN
CURRENT UB PHD STUDENT, FORMER UB UNDERGRAD



JACQUELINE HANNAN
FORMER UB UNDERGRAD



ISYS JOHNSON
CURRENT UB PHD STUDENT



JASON YAN
FORMER UB UNDERGRAD

Come talk to current Ph.D. students at UB and UMich about what you do (and don't) need to do as an undergrad to figure out if you want a PhD and to put yourself in a position to apply!

**15
Nov.,
2022**

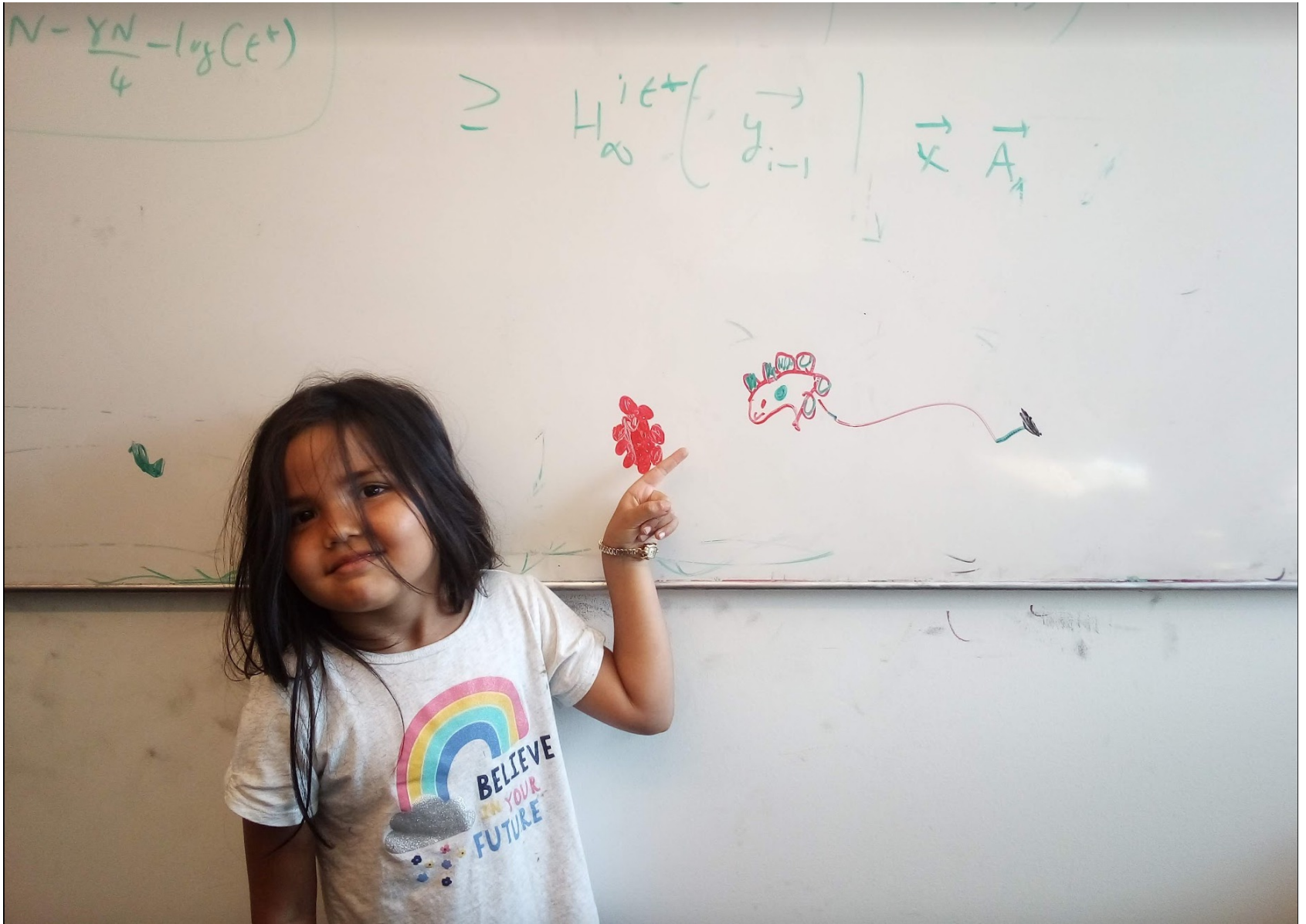
3-4PM

**REGISTER AT
[BIT.LY/PHD_PANEL](https://bit.ly/phd_panel) OR WITH
THE QR BELOW!**



Questions? Email Kenny, josephkena@gmail.com

Questions/Comments?



When to use Dynamic Programming

There are polynomially many sub-problems

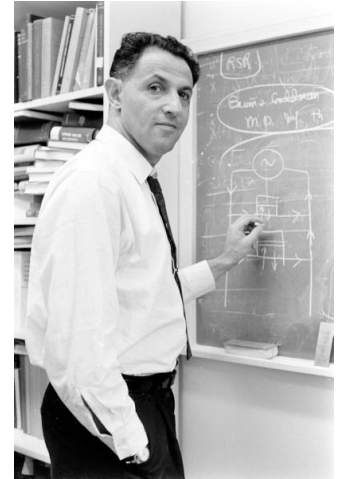
$$\text{OPT}(1), \dots, \text{OPT}(n)$$

Optimal solution can be computed from solutions to sub-problems

$$\text{OPT}(j) = \max \{ v_j + \text{OPT}(p(j)), \text{OPT}(j-1) \}$$

There is an ordering among sub-problem that allows for iterative solution

$$\text{OPT}(j) \text{ only depends on } \text{OPT}(j-1), \dots, \text{OPT}(1)$$



Richard Bellman

Scheduling to min idle cycles

n jobs, i^{th} job takes w_i cycles

You have W cycles on the cloud



What is the maximum number of cycles you can schedule?

Subset sum problem

Input: n integers w_1, w_2, \dots, w_n

bound W

Output: subset S of $[n]$ such that

(1) sum of w_i for all i in S is at most W

(2) $w(S)$ is maximized

Questions?



Today's agenda

Dynamic Program for Subset Sum problem

Algo on the board...

