Lecture 26

CSE 331

Nov 1, 2024

Coding Project 1+2 due TODAY

Fri, Nov 1	Multiplying large integers F23 F22 F21 F19 F18 F17 x^2	[KT, Sec 5.5] (**Project (Problems 1 & 2 `Coding`) in**) *Reading Assignment:* Unraveling the mystery behind the identity
Mon, Nov 4	Closest Pair of Points F23 F22 F21 F19 F18 F17 x^2	[KT, Sec 5.4] (**Project (Problems 1 & 2 `Reflection`) in**)

Get your piazza Qs in by 5pm!

note @266

stop following 0 views

Actions

Piazza response policy reminder

A gentle reminder that as per the piazza response policy listed in the syllabus we cannot guarantee any question posted on piazza after Friday 5pm until Monday 9am.

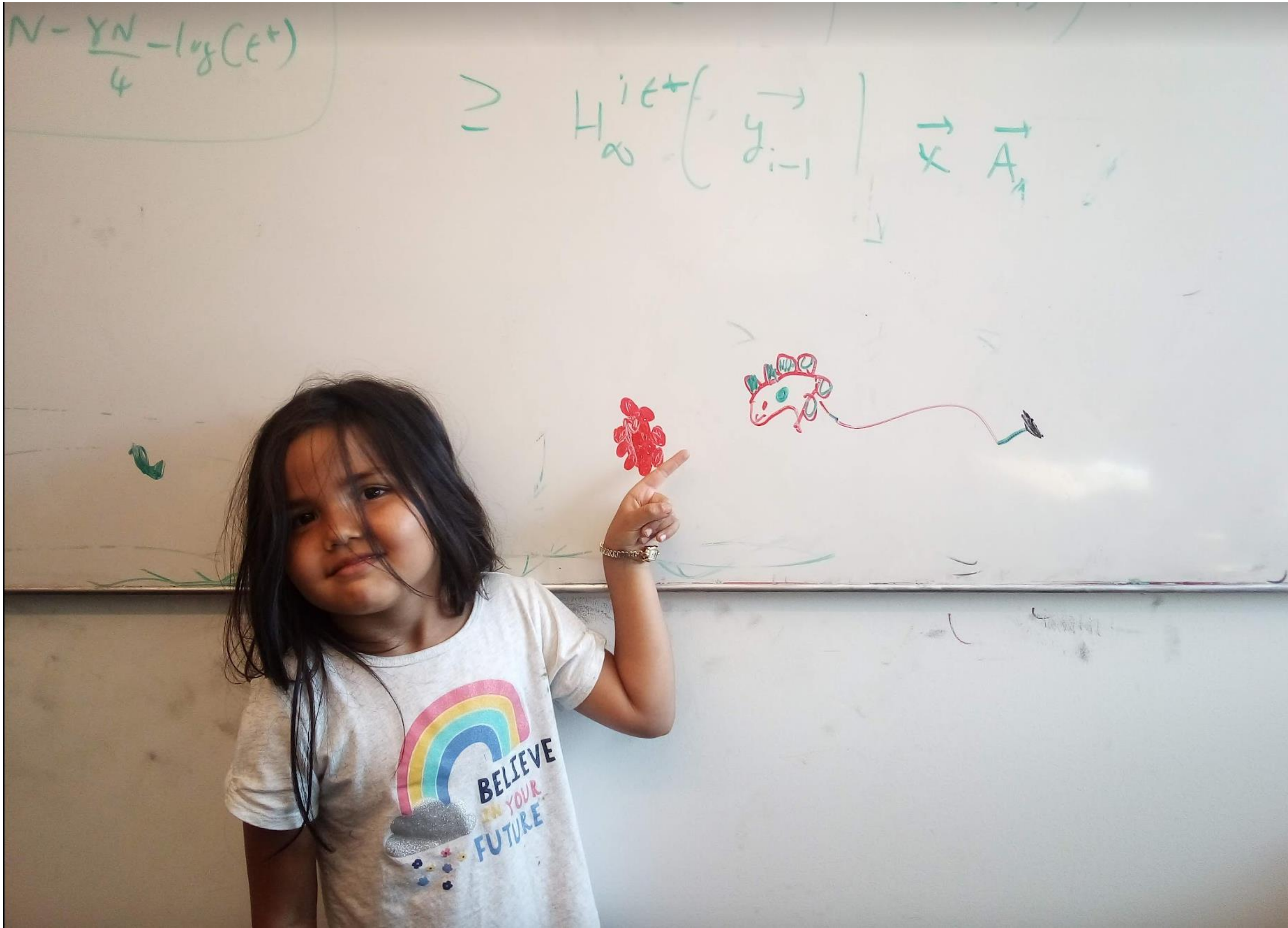
So **if you have project related questions for us, please make sure to get them in before 5pm tomorrow!**

piazza project logistics

Edit good note | 0

Updated 21 seconds ago by Atri Rudra

Questions/Comments?



Divide and Conquer

Divide up the problem into at least two sub-problems

Recursively solve the sub-problems

"Patch up" the solutions to the sub-problems for the final solution

Improvements on a smaller scale

Greedy algorithms: exponential → poly time

(Typical) Divide and Conquer: $O(n^2)$ → asymptotically smaller running time

Multiplying two numbers

Given two numbers a and b in binary

$a=(a_{n-1},..,a_0)$ and $b = (b_{n-1},\ldots,b_0)$

Compute $c = a \times b$

Elementary school algorithm is $O(n^2)$

The current algorithm scheme

Shift by O(n) bits

Adding O(n) bit numbers

Mult over n bits

$$a \bullet b = a^L b^L \bullet 2^{2[n/2]} + (a^L b^R + a^R b^L) \bullet 2^{[n/2]} + a^R b^R$$

Multiplication over n/2 bit inputs

$T(n) \leq 4T(n/2) + cn$

$T(1) \leq c$

T(n) is $O(n^2)$

The key identity

$$a^L b^R + a^R b^L = (a^L + a^R)(b^L + b^R) - a^L b^L - a^R b^R$$

Wait, how do you think of that?

De-Mystifying the Integer Multiplication Algorithm

In class, we saw an $O\left(n^{\log_2 3}\right)$ time algorithm to multiply two n bit numbers that used an identity that seemed to be plucked out of thin air. In this note, we will try and de-mystify how one might come about thinking of this identity in the first place.

The setup

We first recall the problem that we are trying to solve:

> **Multiplying Integers**
> Given two n bit numbers $a = (a_{n-1}, \ldots, a_0)$ and $b = (b_{n-1}, \ldots, b_0)$, output their product $c = a \times b$.

Next, recall the following notation that we used:

$$a^0 = \left(a_{\lceil \frac{n}{2} \rceil - 1}, \ldots, a_0\right),$$

$$a^1 = \left(a_{n-1}, \ldots, a_{\lceil \frac{n}{2} \rceil}\right),$$

The final algorithm

Input: $a = (a_{n-1},..,a_0)$ and $b = (b_{n-1},\ldots,b_0)$

Mult (a, b)

If $n = 1$ return $a_0 b_0$

$a^L = a_{n-1},\ldots,a_{[n/2]}$ and $a^R = a_{[n/2]-1},\ldots, a_0$

Compute b^L and b^R from b

$x = a^L + a^R$ and $y = b^L + b^R$

Let $p =$ Mult (x, y), $D =$ Mult (a^L, b^L), $E =$ Mult (a^R, b^R)

$F = p - D - E$

return $D \bullet 2^{2[n/2]} + F \bullet 2^{[n/2]} + E$

$T(1) \leq c$

$T(n) \leq 3T(n/2) + cn$

$O(n^{\log_2 3}) = O(n^{1.59})$
run time

All green operations are $O(n)$ time

$$a \bullet b = a^L b^L \bullet 2^{2[n/2]} + ((a^L+a^R)(b^L+b^R) - a^L b^L - a^R b^R) \bullet 2^{[n/2]} + a^R b^R$$