Lecture 30

CSE 331

Nov 12, 2021

Please have a face mask on

Masking requirement

UR requires all students, employees and visitors – regardless of their vaccination status – to wear face coverings while inside campus buildings.

https://www.buffalo.edu/coronavirus/health-and-safety/health-safety-guidelines.html

Homework 6 out

Homework 6

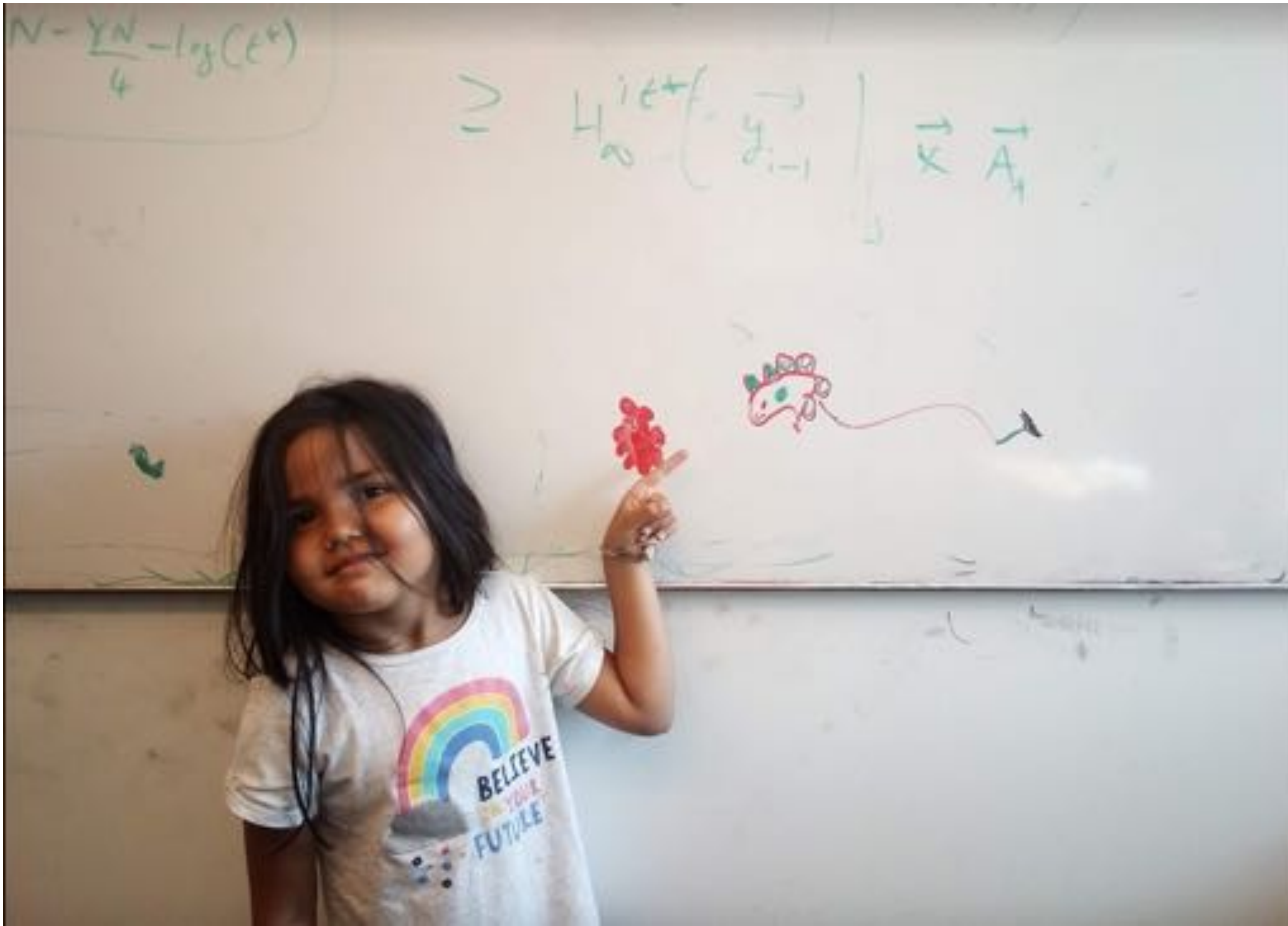
- Part **(b)**: Present a divide and conquer algorithm that given non-negative integers a and n computes `Power` (a, n) in $O(\log n)$ time.

> **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.

Questions/Comments?



Weighted Interval Scheduling

Input: n jobs (s_i, f_i, v_i)

Output: A schedule S s.t. no two jobs in S have a conflict

Goal: max $\Sigma_{i \text{ in } S}\, v_j$

Assume: jobs are sorted by their finish time

Couple more definitions

$p(j)$ = largest $i < j$ s.t. i does not conflict with j

= 0 if no such i exists

$p(j) < j$

$OPT(j)$ = optimal value on instance $1,..,j$

Property of OPT

j in OPT(j)

j not in OPT(j)

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

Given OPT(1),…., OPT(j-1), how can one figure out if j in optimal solution or not?

A recursive algorithm

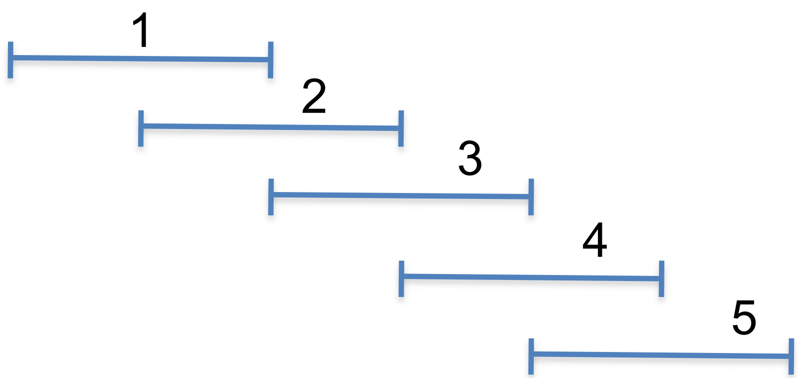
Compute-Opt(j)

```
If j = 0 then return 0

return max { v_j + Compute-Opt( p(j) ), Compute-Opt( j-1 ) }
```

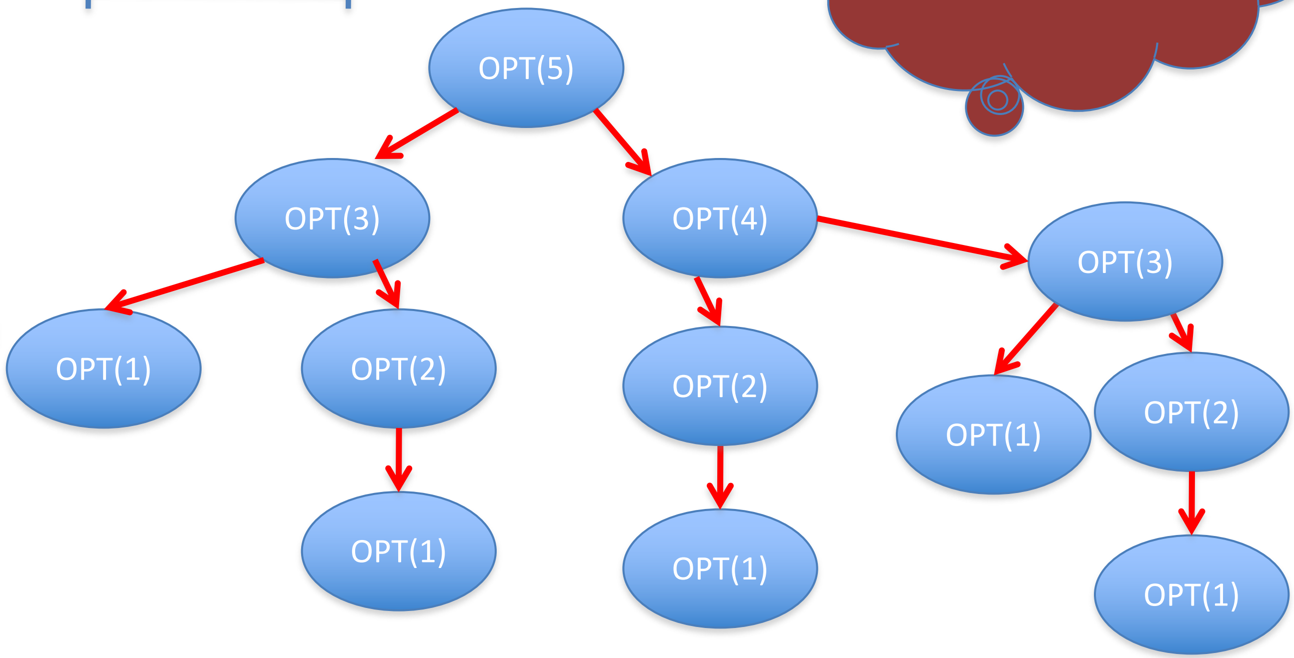
$$OPT(j) = \max \{ v_j + OPT(p(j)), OPT(j-1) \}$$

Exponential Running Time



p(j) = j-2

Only 5 OPT values!



Formal proof: Ex.

A recursive algorithm

M-Compute-Opt(j)

If j = 0 then return 0

If M[j] is not null then return M[j]

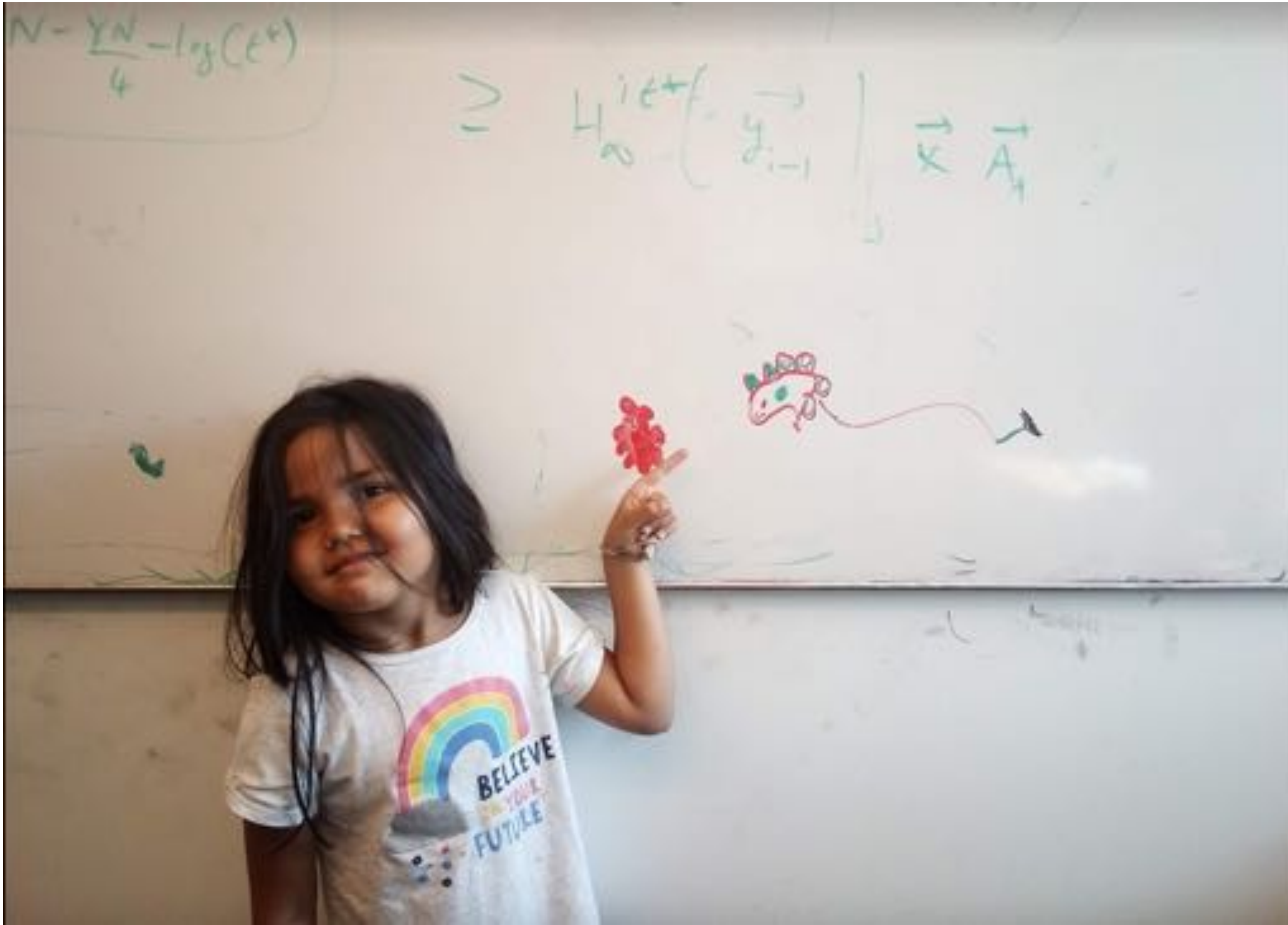
M[j] = max { v_j + M-Compute-Opt(p(j)), M-Compute-Opt(j-1) }

return M[j]

M-Compute-Opt(j) = OPT(j)

Run time = O(# recursive calls)

Questions/Comments?



Bounding # recursions

M-Compute-Opt(j)

If j = 0 then return 0

If M[j] is not null then return M[j]

M[j] = max { v_j + M-Compute-Opt(p(j)), M-Compute-Opt(j-1) }

return M[j]

O(n) overall

Whenever a recursive call is made an M value is assigned

At most n values of M can be assigned

Property of OPT

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

Given OPT(1), …, OPT(j-1),
one can compute OPT(j)

Recursion+ memory = Iteration

Iteratively compute the OPT(j) values

Iterative-Compute-Opt

M[0] = 0

For j=1,…,n

M[j] = max { v_j + M[p(j)], M[j-1] }

M[j] = OPT(j)

O(n) run time

Algo run on the board…

Reading Assignment

Sec 6.1, 6.2 of [KT]

When to use Dynamic Programming

Richard Bellman

There are polynomially many sub-problems

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

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

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

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?

Rest of today’s agenda

Dynamic Program for Subset Sum problem

May the Bellman force be with you